VŠB – Technická univerzita Ostrava
Fakulta elektrotechniky a informatiky
Katedra informatiky

Multiplatformní aplikace s využitím webových technologií

Multiplatform App Based on Web Technologies

2018 Bc. Radovan Macháček

VŠB - Technická univerzita Ostrava
Fakulta elektrotechniky a informatiky
Katedra informatiky

Zadání diplomové práce

Student: **Bc. Radovan Macháček**

Studijní program: N2647 Informační a komunikační technologie

Studijní obor: 2612T025 Informatika a výpočetní technika

Téma: Multiplatformní aplikace s využitím webových technologií
Multiplatform App Based on Web Technologies

Jazyk vypracování: čeština

Zásady pro vypracování:

Cílem práce je realizovat aplikace pokrývající současné možnosti vývoje aplikací pro různé mobilní platformy, a to s využitím webových technologií. Práce poskytne přehled o současných možnostech a technologiích a zároveň také případové studie implementace různých přístupů.

1. Zmapujte oblast multiplatformních prostředků (např. Xamarin, React, Cordova) pro vývoj mobilních aplikací s využitím webových technologií.
2. Popište základní koncepty vývoje takovýchto aplikací.
3. Popište webové technologie využité při vývoji, a to s ohledem na jejich rozšíření a specifické využití ve vazbě na konkrétní framework.
4. Navrhněte ukázkovou aplikaci, která bude ilustrovat rozšířené možnosti multiplatformních aplikací, např. využití fotoaparátu, GPS lokalizace, lokální zdrojů.
5. Implementujte ukázkovou aplikaci nejméně ve dvou multiplatformních vývojových prostředcích.
6. Zhodnoťte výsledné aplikace a jejich praktické nasazení.

Seznam doporučené odborné literatury:

[1] Jon Duckett: JavaScript and JQuery: Interactive Front-End Web Development, Wiley, 2014, ISBN: 978-1118531648
[2] Sasha Vodnik: HTML5 and CSS3, Illustrated Complete, Course Technology, 2015, ISBN: 978-1305394049
[3] Jason Beaird: The Principles of Beautiful Web Design, SitePoint, 2014, ISBN: 978-0992279448
[4] Erixc Elliot: Programming JavaScript Applications: Robust Web Architecture with Node, HTML5, and Modern JS Libraries, O'Reilly Media, 2014, ISBN: 978-1491950296

Formální náležitosti a rozsah diplomové práce stanoví pokyny pro vypracování zveřejněné na webových stránkách fakulty.

Vedoucí diplomové práce: **Ing. Michal Radecký, Ph.D.**

Datum zadání: 01.09.2017

Datum odevzdání: 30.04.2018

doc. Ing. Jan Platoš, Ph.D.	prof. Ing. Pavel Brandštetter, CSc.
vedoucí katedry	*děkan fakulty*

Prohlašuji, že jsem tuto diplomovou práci vypracoval samostatně. Uvedl jsem všechny literární prameny a publikace, ze kterých jsem čerpal.

V Ostravě 30. dubna 2018 .....

Souhlasím se zveřejněním této diplomové práce dle požadavků čl. 26, odst. 9 Studijního a zkušebního řádu pro studium v magisterských programech VŠB-TU Ostrava.

V Ostravě 30. dubna 2018 .....

Rád bych na tomto místě poděkoval panu doktoru Michalu Radeckému za užitečné rady a pomoc během vypracování této diplomové práce. Dále bych rád poděkoval Šimonu Dolenskému za zapůjčení mobilního telefonu pro testovací účely.

Abstrakt

Tato práce se zabývá problematikou multiplaformního vývoje založeném na webových technologiích. Hlavní myšlenkou je popis a porovnání různých multiplatformních frameworků a jejich základních principů. K naplnění této myšlenky byly vytvořeny ukázkové aplikace, které implementují rozšířené možnosti multiplatformního vývoje.

Klíčová slova: multiplatfromní vývoj, mobilní aplikace, PhoneGap, React Native, Rozšířená realita

Abstract

This thesis deals with issues of multiplatform development based on web technologies. The main idea is to describe and compare different multiplatform frameworks and their basic principles. To meet this idea, sample applications have been developed that implement enhanced multiplatform development capabilities.

Key Words: multiplatform development, mobile application, PhoneGap, React Native, Augmented Reality

Obsah

Seznam použitých zkratek a symbolů

AR	–	Augmented Reality
API	–	Application Programming Interface
CC	–	Cross-Compiled
CLI	–	Command-line Interface
CSS	–	Cascading Style Sheet
DOM	–	Document Object Model
GPS	–	Geolocation Positioning System
GUI	–	Graphical User Interface
HTML	–	Hyper Text Markup Language
IDE	–	Integrated Development Environment
IL	–	Intermediate Language
JSON	–	JavaScript Object Notation
JSX	–	JavaScript XML
LLVM	–	Low Level Virtual Machine
MVC	–	Model - View - Controller
RVC	–	Reality-Virtuality Continuum
WWDC	–	Worldwide Developers Conference
UIX	–	User Interface Experience
XAML	–	Extensible Application Markup Language

Seznam obrázků

Seznam tabulek

Seznam výpisů zdrojového kódu

1 Úvod

V posledních několika letech zažívá prodej chytrých telefonů strmý růst. Zlepšující se hardwarové a softwarové vybavení nemá přílišný vliv na cenovou hladinu a může si je dnes pořídit široká veřejnost. V roce 2016 bylo společností Statcounter sledováno 2,3 milionů webových stránek a více než 51% načtení webové stránky proběhlo na mobilu nebo tabletu. V roce 2010 to bylo pouze 10% a v roce 2013 46,5% [3]. Pro člověka je chytrý telefon nezbytnou součástí každodenního života.

Dnes existují aplikace na téměř všechny služby, od objednání hotelového pokoje až po ovládání kávovaru, který stiskem tlačítek začne připravovat šálek kávy. Mobilní zařízení se staly centrálním prostředkem k vykonávání lidské činnosti.

Vzhledem k tomu, že na trhu chytrých telefonů dominují vzájemně odlišné platformy, musí vývojáři čelit rozhodnutí, na jakou z nich se zaměřit. Multiplatformním přístupem můžeme vytvářet aplikace pro oba systémy současně, a snížit tak rychlost vývoje, a tím pádem i jeho cenu. Bohužel, někdy na úkor kvality.

Trh s mobilními aplikacemi je rozdělen na dva dominující operační systémy, jež dohromady pokrývají přes 90% všech telefonů [2]. Hlavním cílem společností, které se zaměřují na multiplatformní vývoj, je tedy získat, co největší počet uživatelů aplikace, s co nejmenšími náklady na vývoj. V ideálním případě to znamená vyvíjet aplikaci najednou pro Android i iOS. Podstatné v tomto případně nejsou jenom náklady a čas, ale také z dlouhodobého hlediska také udržitelnost.

V úvodní části se práce věnuje různým přístupům k multiplatformnímu vývoji a popisem základních pojmů týkajících se mobilních operačních systémů. Dále se práce zabývá kritérii pro volbu vhodného nástroje umožňujícího multiplatformní vývoj. Následovně autor popisuje vlastnosti a architektury vybraných multiplatformních frameworků.

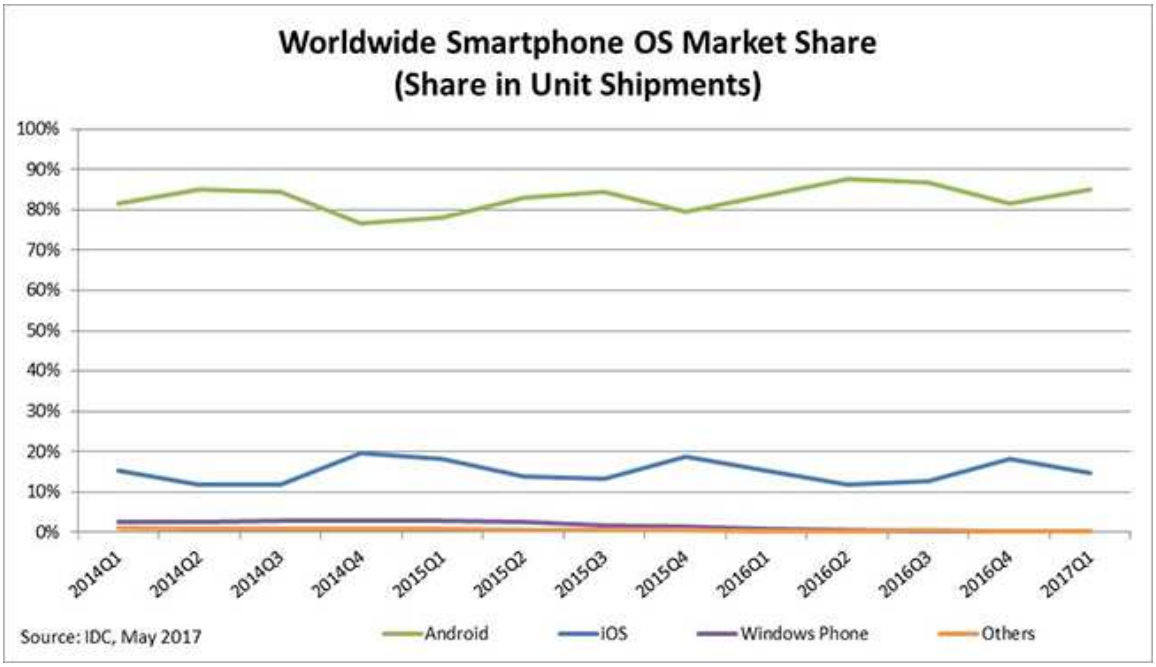
Poté je představena referenční aplikace, vytvořená v nástroji PhoneGap, která má za úkol implementovat rozšířené možnosti multiplatformních aplikací. V závěru je představena další vytvořená aplikace, sloužící pro porovnání vlastností a procesů vývoje ve frameworku PhoneGap a React Native.

2 Mobilní operační systémy

Okolo roku 2000 byl mobilní telefon téměř v každé domácnosti. Jeho hlavním využití se týkalo posílání SMS zpráv a volání. Pouze malé procento zařízení dovolovalo uživateli přistupovat k internetu nebo využít jiné rozšířené funkce.

Tento typ telefonu se často využíval v obchodním a podnikatelském prostředí. Operační systémem těchto zařízení byl převážně BlackBerry OS nebo Windows CE, z nichž se posléze vyvinuly pokročilejší operační systémy smartphonů. Prvním dominujícím systémem na trhu se stal Symbian OS. Velké společnosti viděly potenciál v kombinaci možností počítače a mobilního telefonu do jednoho zařízení.

V roce 2007 měl Symbian prvního konkurenta s názvem iOS a o rok později vyšel další rival Android. Zájem o Symbian posléze prudce upadl a trh v roce 2010 změnil své rozvržení. Dominující role převzali Android s iOS. Windows Phone poté zabíral menší procento trhu, ale jeho podpora byla posléze ukončena a jeho zastoupení ještě významně oslabilo [4].



Obrázek 1: Zastoupení mobilních operačních systémů na trhu od roku 2014[2]

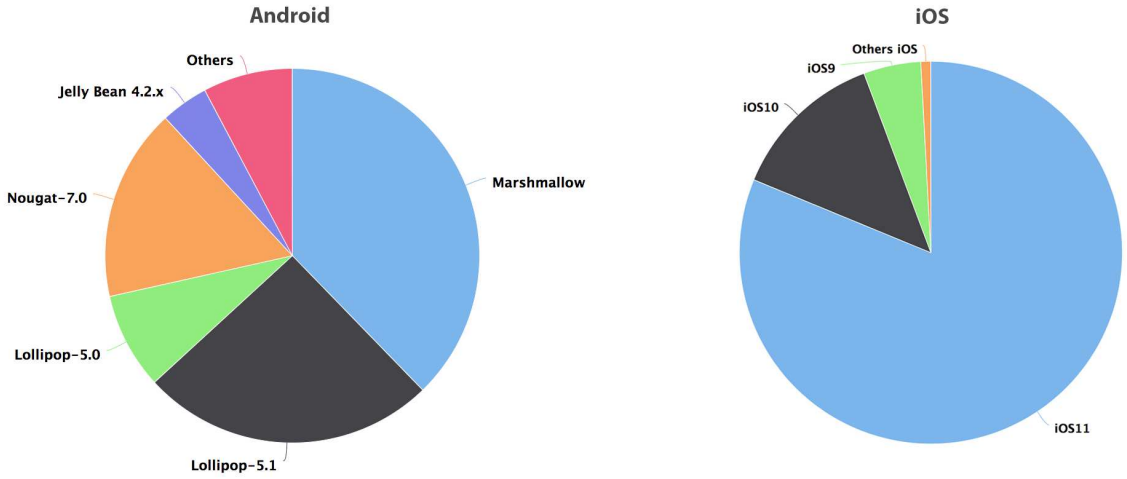
2.1 Android

Operační systém Android vyvíjí společnost Google. Systém je založený na linuxovém jádru vydaném jako open-source software. Platforma je cílově zaměřena pro mobilní telefony a tablety, ale můžeme ji najít i v chytrých televizích. Android několik let drží prvenství v zastoupení smartphonů na trhu. Převážná většina výrobců mobilních zařízeních si Android upravuje vlastní vrstvou softwaru. Z tohoto důvodu dochází k výraznější fragmentaci verzí na trhu. Starší zařízení

nemusí podporovat novější verze Androidu nebo výrobce mobilního telefonu nemá upravenou verzi přímo systému pro svůj produkt.

2.2 iOS

Operační systém iOS byl poprvé představen světu v roce 2007 na Macworld konferenci. Spolu s ním by uveden i první chytrý telefon společnosti Apple s názvem iPhone. Platforma iOS se stala více uzavřenou oproti Androidu. Počet zařízení podporující iOS je málo a všechny produkuje jediná společnost. Jedna z domén firmy Apple je bezpečnost, z toho důvodu nejsou data sdíleny na všechna zařízení od jednoho Apple ID. Dále aplikace, jenž jsou přidávány na Apple store podléhají pečlivé kontrole. Před uvedením aplikace do obchodu, ji pracovník spustí a zkontroluje, aby zjistil, zda-li vyhovuje všem podmínkám a bezpečnostním protokolům.



Obrázek 2: Poměr verzí operačních systémů Android a iOS v roce 2017 [16][17]

2.3 Uživatelské rozhraní Android a iOS

Uživatelské rozhraní patří mezi velmi důležité fáze návrhového procesu. Hlavní funkce aplikace se zde převedeny do grafické podoby a správně navržené uživatelské rozhraní pomáhá uživateli intuitivně ovládat aplikaci. Obě platformy přistupují ke grafického rozhraní rozličným způsobem. Apple poskytuje vývojářům aplikací stručnou filozofii jakými směry se má uživatelské rozhraní ubírat. Nevyzdvihuje konkrétní designový styl nebo téma, ale zdůrazňuje spíše obecné principy, kterými se řídit.

Rady pro vytvoření uživatelského rozhraní pro iOS [31]:

- Zřetelnost - text musí být čitelný v každé velikosti a v každém rozlišení. Ikony by měly být zpracovány precizně a měly by podtrhovat funkčnost vzhledu.

- Integrita - uživatelské rozhraní využívá již zaběhlých komponent. Uživatel není zmaten a dokáže předvídat chování jednotlivých prvků.
- Použití metafor - uživatel je obeznámen s prvky virtuálních objektů a na základě zkušenosti dokáže aplikaci jednoduše ovládat.
- Neustálá zpětná vazba - aplikace by pro každou akci uživatele měla poskytovat srozumitelnou odpověď. Například při zpracovávání nějaké operace, by měl být o procesu informován.
- Ovládání uživatele - uživatel by se měl cítit, že on je hlavním aktérem aplikace. Aplikace by měla mít určitou rovnováhu mezi svobodou uživatele a vlastní konfigurací, aby nedocházelo k nežádoucím výstupům. Před vykonáním radikálních akcí by měl být uživatel upozorněn.
- Přímá manipulace - veškerý obsah na obrazovce má být pro uživatele srozumitelný. Je velmi doporučeno používání gest pro interakci s objekty a po každé akci by měla být ihned viditelná změna, kterou uživatel provedl.

Android volí trochu odlišnou strategii. Na webových stránkách pro vývojáře je podrobný návod, jak jednotlivé složky uživatelského rozhraní vytvářet. Od celkového rozvržení až po jednotlivé prvky.

Android razí určitý vzhled uživatelského rozhraní, nazývající se Material design. Hlavní rysy Material tvoří zejména vizuální jednoduchost jednotlivých prvků a minimální prostorová koncepce rozhraní. Material byl vytvořen za účelem sjednocení UI do jednotného návrhového systému.

2.4 Windows Phone

V roce 2004 začala společnost Microsoft pracovat na mobilním operačním systému pod názvem Photon, jenž měl být během roku 2007 spuštěn. Spuštění bylo však několikrát odloženo. V roce 2007 vyšel iPhone, jenž vnesl úplně jiný pohled na dosavadní modely mobilních telefonů a nastartoval éru smartphonů. Photon byl poté přejmenován na Windows Mobile a v roce 2010 na Windows Phone.

Windows Phone překvapil hlavně svou designovou filozofií Metro design. Microsoft poté uzavřel partnerství s Nokia, která vyměnila dosavadní operační systém na svých mobilních telefonech Symbian právě za Windows Phone.

Později v roce 2014 byla vydána poslední verze Windows Phone 8.1. V následujícím roce Microsoft vytvořil nový operační systém společný pro mobilní zařízení i počítače s názvem Windows 10. Tímto krokem Microsoft finálně podporu a podporu Windows Phone. [18]

3 Vývoj multiplatformních mobilních aplikací

Hlavním cílem multiplatformního vývoje, je napsat v nejlepším případě jeden zdrojový kód, který bude spustitelný na co největším množství operačních systémů. Pokud se podaří splnit paradigma "write once - run anywhere", odpadne tím potřeba několika vývojových prostředí a programovacích jazyků, což může v konečném důsledku ušetřit čas a samotné náklady na aplikaci.

Možnost vývoje multiplatformních aplikací existují již delší dobu, ale za posledních pár let se významně zdokonalila, zejména díky zájmu velkých společností, podílejících se na vývoji frameworků pro vývoj mobilních aplikací.

Nyní se nabízí pět přístupů určených pro vývoj multiplatformní aplikace:

- nativní aplikace
- hybridní aplikace
- webová aplikace
- interpretovaná aplikace
- aplikace kompilovaná křížovým překladačem

3.1 Nativní aplikace

Nativní způsob vývoje aplikací je poměrně rozšířený a zejména díky fragmentaci trhu nastává otázka, zda-li je tento způsob pro společnosti nadále ekonomicky efektivní. Nevýhoda tohoto přístupu spočívá v naprogramování aplikace pro každou platformu zvlášť. Velké společnosti, které mají dostatek zdrojů, to nemusí být moc složité, ale pro malé subjekty, jenž jsou soustředěny na jeden programovací jazyk, to může znamenat značný problém.

Na druhou stranu, obrovskou výhodou nativních aplikací jsou distribuční obchody specifické pro každou platformu (Google Play a App Store). Dále také robustnost a vysoká spolehlivost. Nativní aplikace mají výhodu v přímém přístupu na hardware zařízení a díky tomu mohou být podstatně rychlejší a plynulejší. [10]

3.2 Hybridní aplikace

Hybridní aplikace vytvářejí jakýsi pomyslný most mezi nativním a webovým přístupem. Podobají se webovým aplikacím, poněvadž často využívají základní webové technologie, jenž jsou implementovány prohlížeči. Prostředí prohlížeče je poté zabaleno do nativní aplikace, aby mohlo být spuštěno nad daným operačním systémem.

Vývoj takových aplikací je poměrně levný a rychlý, kvůli základním vývojovým technologiím jako jsou HTML, CSS a JavaScript. Pro spuštění na různých platformách se přepisuje pouze

minimální část nativního kódu. Aplikace může nést všechny výhody nativního prostředí, týkajících se distribuce na obchodech jako Google Store nebo App Store. Na druhou stranu hybridní aplikace nedosahují stejné rychlosti jako nativní, protože závisí na rychlosti prohlížeče. Jako další nevýhodu je třeba brát v potaz i prožitek uživatele, pro kterého nemusí aplikace vypadat esteticky tak dobře jako nativní aplikace.

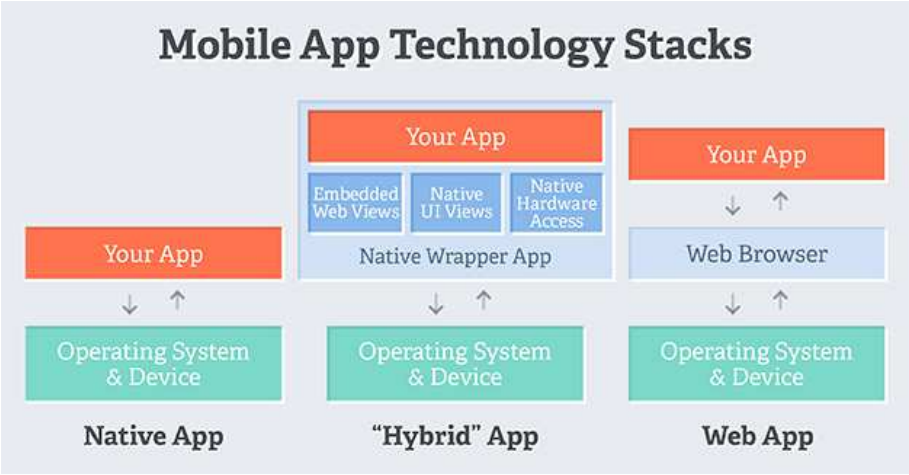
Hybridní aplikace nemá přístup k nativním prvkům uživatelského rozhraní, což může být výhoda i nevýhoda, jelikož aplikace bude mít téměř identické UI na všech platformách.[13]

3.3 Webové aplikace

Webové aplikace jsou v podstatě webové stránky uzpůsobené pro zobrazení na mobilním telefonu. Mohou se v mnoha případech podobat nativním aplikacím, protože dnes můžeme přes webový prohlížeč přistupovat k některým částem hardwaru.

Vzhledem k tomu, že jsou postaveny na standardních webových technologiích, musejí být spuštěny ve webovém prohlížeči. Spuštění probíhá jako při návštěvě webové stránky pomocí URL. Vývoj takové aplikace je velice efektivní, co se týče ceny a času. Stačí napsat pouze jeden kód spustitelný na všech platformách, a nebude vyžadovat téměř žádné místo v paměti zařízení. Nevýhodou tohoto přístupu je ladění pro různé rozlišení, neboť se aplikace nemusí zobrazovat úplně identicky na různých zařízeních. Odstranění tohoto problému zabere podstatně méně času, než kdyby byla aplikace vytvořena od začátku v jiném programovacím jazyce, jako je tomu u nativního přístupu. [13]

Na níže uvedeném obrázku můžeme vidět strukturu jednotlivých přístupů. Nativní aplikace komunikuje přímo s operačním systémem zařízení. Webová aplikace potřebuje webový prohlížeč jako mezivrstvu nebo rozhraní pro operační systém. Hybridní aplikace má v sobě zabudovaný prohlížeč, jenž je obalen do nativní aplikace.



Obrázek 3: Struktury přístupů pro vývoj mobilní multiplatformní aplikace [1]

3.4 Intepretované aplikace

Interpretované aplikace využívají běh virtuálního prostředí nebo stroje, a zároveň jsou propojeny s nativní vrstvou. Na poli tohoto přístupu, působí výhradně frameworky určené pro vývoj v jazyce JavaScript, podobně jako u hybridních a webových aplikací. Interpretovaný přístup nevyužívá pro zobrazování webový prohlížeč, tudíž je podstatně rychlejší a přístup k API senzorům a dalším možnostem nativního rozhraní je téměř bez omezení. [13]

Při změně kódu nevyžadují opětovné sestavení, proto je možné aplikace vyvíjet s hot reload. Hot reload zlepšuje vývojový proces, poněvadž zavadí nové verze souborů, jenž jsou v rámci běhu aplikace změněny. To přináší mnoho výhod, například udržení stavu aplikace, v jakém se vývojář nachází, nebo rychlejší ladění uživatelského rozhraní [15].

3.5 Aplikace kompilované křížovým překladačem

Cross-compiler, nebo-li křížový překladač, překládá zdrojový kód do nativního kódu dané aplikace nebo knihovny. Jedná se o komplikovanější proces sestavování, než je tomu u interpretovaných aplikací. Tento přístup, lze uplatnit u všech vrstev aplikace, od datové až po uživatelské rozhraní a dosáhnout tak paradigmatu "write once - run anywhere". Křížová kompilace je poměrně složitý proces a některé nástroje jej kombinují s interpretovaným přístupem. Takovým způsobem sestavené aplikace jsou rychlostí velice blízko nativním.

4 Frameworky pro multiplatformní vývoj

Dnes existuje několik desítek různých nástrojů a služeb pro vytváření multiplatformních mobilních aplikací. Pro vybrání vhodného vývojového nástroje je třeba brát v potaz několik důležitých faktorů. Jednou z hlavních myšlenek této práce je porovnat vývojové nástroje pro multiplatformní mobilní aplikace se zaměřením na jejich rozšířené možnosti využití přístupu k nativnímu API. Pro zmenšení počtu prozkoumaných frameworků bylo vybráno několik kriterií, která by měly obecně zastupovat vlastnosti aktivně rozvíjených nástrojů pro multiplatformní vývoj a jsou vhodné pro implementaci referenční aplikace.

- Nástroj je stále vyvíjen - to znamená, že kolem daného frameworku existuje aktivní komunita lidí, která se jej snaží rozvíjet směrem k aktuálním verzím podporovaných operačních systémů. Vývoj jde neustále dopředu a operační systémy na mobilních telefonech jsou aktualizovány v krátkém časovém intervalu. Daný nástroj se tedy musí přizpůsobovat nejnovějším verzím systémů.
- Podporuje Android a iOS - pro pokrytí, co nejvíce zařízení na trhu s mobilními telefony, je potřeba, aby umožňoval vývoj aplikace pro nejpopulárnější mobilní operační systémy. To zaručuje jeho obecnou použitelnost pro většinu zařízení.
- Umožňuje použití nativního API - Musí být dovoleno přistupovat k datům z GPS, magnetometru, gyroskopu a kameře a dalším hardwarovým prvkům. Některé frameworky jsou limitovány pouze rozhraním webového prohlížeče a nemohou tudíž nijak interagovat s některým API hardwaru na mobilního telefonu.
- Hlavním účelem není vývoj her - Přestože existuje několik multiplatformních frameworků pro vývoj her. Cílem nástroje nemá být vytvoření aplikace jako hry, ale obecně používané aplikace využívající nativní API.

Tabulka 1: Vlastnosti osmi vybraných frameworků

Název	Jazyk	Typ aplikace	Podporované platformy
Codename One	Java	Interpretovaný	iOS, Android, Windows
Ionic	JavaScript, AngularJS	Hybridní	iOS, Android, Windows
NativeScript	JavaScript, AngularJS	Interpretovaný	iOS, Android, Windows
PhoneGab	Javascript	Hybridní	iOS, Android, Windows
React Native	JavaScript, React	Interpretovaný	iOS, Android
RubyMotion	Ruby	CC	iOS, Android
Kivy	Python	CC	iOS, Android, Windows
Xamarin	C#	CC	iOS, Android, Windows

Pro širší popis byl vybrán vzorek populárních frameworků se zastoupením jednotlivých multiplatformních přístupů. Za hybridní přístup byl zvolen PhoneGap. Skupinu frameworků, které

využívají křížový kompilátor zastupuje Xamarin a jako poslední React Native, využívající k vytvoření aplikace interpretaci jazyka.

4.1 PhoneGap

Jeden z nejpopulárnějších nástrojů pro tvorbu multiplatformních aplikací je PhoneGap, podle serveru graph.uk. [19] Poprvé se dostal do podvědomí v roce 2009, kdy jej vytvořila firma Nitobi. Poté byl v roce 2011 koupen společností Adobe a nyní jeho vývoj probíhá pod záštitou neziskové organizace Apache Foundation. Zdrojový kód PhoneGab posloužil jako základní kámen pro open-source projekt Apache Cordova. [20]

PhoneGap má podrobně a přehledně zpracovanou dokumentaci, v níž si vývojář může přečíst základní příkazy s tutoriálem pro vývoj aplikace. Velkou výhodou PhoneGap je množství podpory pluginů třetích stran. Jelikož Apache Cordova a PhoneGap jsou postaveny na společné codebase, mohou vzájemně sdílet databázi pluginů.

Komunita okolo frameworku PhoneGap patří mezi nejpočetnější na poli multiplatformního mobilního vývoje. Adobe pro udržování živé a početné komunity pořádá pravidelné internetové konference na aktuální téma. PhoneGap není jedinou distribucí Apache Cordova. Například Ionic, který dovoluje vývoj v AngularJS a zaměřuje se více na uživatelské rozhraní a UX.

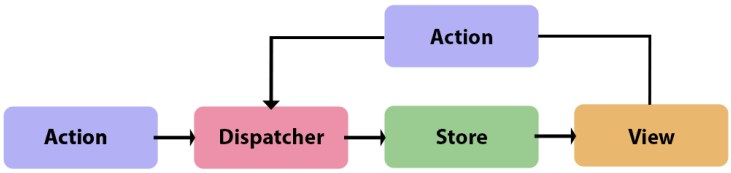
4.2 React Native

React native byl vytvořen společností Facebook. Architektura frameworku vychází z webového frameworku ReactJS, ale překlápí jej do vývoje aplikací pro Android a iOS. Interpretovaný framework využívá alternativní reprezentaci uživatelského rozhraní podobně jako nástroje pro hybridní aplikace. Přístup React Native zde byl již dříve, zastupován nástrojem Appcelerator. Kvůli klesající popularitě PhoneGap, jsou interpretované nástroje jako NativeScript a hlavně React Native na vzestupu. [22]

React Native je v intenzivním vývoji a nové verze vycházejí v poměrně krátkých intervalech, to má za následek, že některé pluginy třetích stran jsou velice často neaktuální, a vzhledem k tomu, že jsou vyvinuty pro konkrétní verzi nebo za jedním účelem, ve vyšších verzích React Native nefungují. Programátor poté musí pro určité funkcionality psát nativní kód zároveň na obě platformy, a to může prodloužit čas vývoje.[21]

4.2.1 Flux

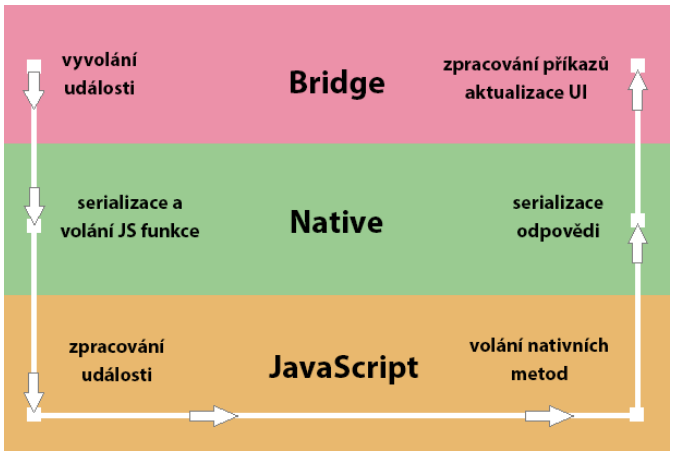
Flux je návrhový vzor používaný společností Facebook pro tvorbu uživatelského rozhraní webových aplikací. Konkrétně Facebook tento návrhový vzor využívá pro práci s frameworkem React. Hlavním úkol vzoru spočívá ve zpracování toku dat. Vychází z konceptu, že každé uživatelské rozhraní je tvořeno daty, které se posílají z uložišť (Store). Akce uživatele poté zpracovává Dispatcher.[32]



Obrázek 4: Model architektury Flux

4.2.2 Bridge

Bridge asynchronně zprostředkovává komunikaci mezi nativní vrstvou aplikace a JavaScriptovou vrstvou. Na obrázku níže, jsou znázorněny jednotlivé kroky při vyvolání události. K samotnému vyvolání dochází v nativní vrstvě, a to například dotekem obrazovky nebo přímo zavoláním konkrétní systémové funkce. Poté se přechází k notifikaci JavaScriptu přes Bridge, nabízející serializované rozhraní pro komunikaci. Následně je událost zpracována a vyhodnocena. Po vyhodnocení jsou zavolány metody z nativní vrstvy. V posledním bodu, Bridge předá příkazy z JavaScriptu, a pokud je potřeba, nativní vrstva aktualizuje uživatelské rozhraní.[27]



Obrázek 5: Zjednodušený model komunikace React Native

4.3 Xamarin

Xamarin je vývojářská firma produkující stejnojmenný multiplatformní mobilní framework od roku 2011. O pět let později jej poté akceptoval Microsoft pro vývoj multiplatformních aplikací, tím že jej integroval do svého vývojového nástroje Vistual Studio. Platforma Xamarin zpro-

středkovává programátorovi prostředí pro napsáni aplikace v jazyce C# najednou pro Windows, Android i iOS. [24]

Vývojáři mohou používat Xamarin Studio nebo Visual Studio, ale pouze iOS aplikace mohou být sestaveny na zařízení s Mac OS s IDE Xcode [12]. Xamarin staví na projektech Mono for Android a MonoTouch, mapující nativní API mobilních operačních systémů Android a iOS.

Pro kompilaci na platformu iOS Xamarin využívá ahead-of-time kompilaci na rozdíl od systému Android, který kompiluje do IL. Následně je IL zabalen do Mono Virtual Machine. Ahead-of-time kompilace zahrnuje kompilaci celého .NET frameworku i s nepoužitými třídami. Tyto třídy jsou poté během linkování odstraněny, aby se redukována velikost aplikace. U Intermediate Language aplikace běží na pozadí Java/ART, což je v podstatně Android runtime, jenž pomocí Java Native Interface komunikuje s nativní vrstvou. Nevyužité třídy jsou opět vyřazeny během linkovacího procesu.[25]

4.3.1 Xamarin Forms

Xamarin Forms je vlastní nástroj frameworku Xamarin pro tvorbu uživatelského rozhraní. Každá platforma má specifické prvky a layouty, na které je nutno namapovat adekvátní prvky rozhraní. Vzhledem k tomu, že nabídka je poměrně velká, může si vývojář velice rychle vytvořit uživatelské rozhraní pro podporované platformy. Jako značkovací jazyk pro UI se většinou používá XAML.

Custom Renderer dokáže změnit vykreslování XAML deklarace ovládacích prvků uživatelského rozhraní. Upravuje funkcionalitu i vzhled zvlášť pro každou platformu. Lze tedy využít specifické funkce pro uživatelské rozhraní dané platformy [23].

5 Referenční aplikace s rozšířenou realitou

Tato kapitola popisuje vybrané technologické části pro vývoj referenční aplikace SmogAR. Jako demonstrační prvek pro rozšířené možnosti mobilních zařízení byla zvolena implementace rozšířené reality, využívající kameru a několik dalších senzorů zařízení. Aplikace má ukázat využití rozšířených možností při multiplatformním vývoji.

5.1 Popis aplikace rozšířené reality

Referenční aplikace bude zprostředkovává pohled skrze rozšířenou realitu obohacenou o data z Internetu. Například informace o různých zeměpisných bodech, na které se uživatel skrze kameru zrovna dívá. Uživateli bude umožněno nastavení různých atributů virtuální reality. Virtuální překrytí kamery bude také interaktivní a uživatel se po kliknutí dostane na detailnější informace.

5.2 Rozšířená realita

V roce 1994 Paul Milgram a Fumio Kishino definovali rozšířenou realitu jako něco mezi virtuální realitou a reálným světem. Vytvořili model Reality-Virtuality Continuum. Model určuje úroveň virtuality v prostředí. Na obrázku číslo 6 můžeme pozorovat nákres RVC. Na okrajích se nachází realita a virtualita. Mezi nimi je možné pozorovat smíšenou realita, jenž se následně dělí podle toho, zda je blíže reálnému nebo virtuálnímu obrazu na rozšířenou realitu nebo rozšířenou virtualitu [9].

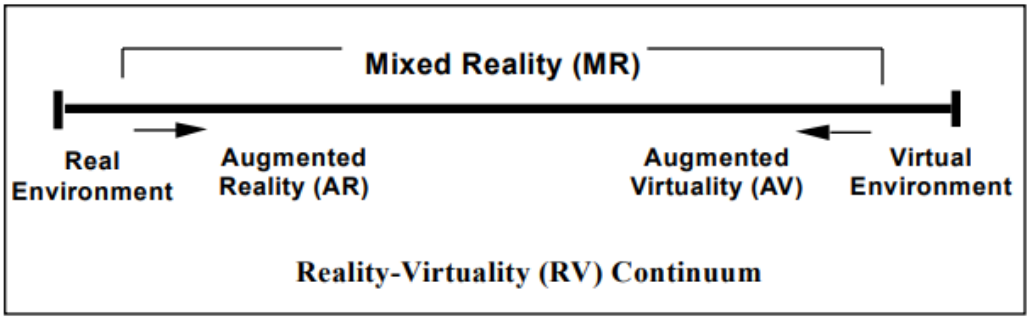
5.2.1 Virtuální realita

Za virtuální realitu považujeme počítačem vygenerovaný obraz reálného světa [8]. To znamená, že se jedná o celkovou umělou simulaci prostředí a probíhajících dějů. Zařízení zobrazující virtuální realitu jsou v podstatě brýle, které pro každé oko vykreslují virtuální obraz prostředí a pomocí gyroskopu a dalších senzorů dokáží snímat polohu uživatele, polohu hlavy a rotaci těla.

První zařízení nositelné na hlavě umožňující velice jednoduchou virtuální realitu, vytvořil Ivan Sutherland v roce 1968. Poté až v roce 2014 se technologie dostala do širšího povědomí uživatelů, zejména díky společnosti Google, která představila Google Cardboard. Toto zařízení velice levně, jednoduše a za použití mobilního telefonu zprostředkovalo virtuální realitu pro široké spektrum uživatelů [26].

Rozšířená realita je ve své podstatě obraz světa, rozšířený o vrstvu počítačem doplněných objektů. Obraz reálného světa z videokamery nebo fotoaparátu je expandován o úroveň, zobrazující 3D nebo 2D objekty. Na virtuální vrstvě se mohou objevit dvojrozměrné štítky s informacemi nebo 3D modely zastupující reálné předměty.

Virtuální realita silně závisí na hardwarovém vybavení zařízení. Mobilní telefony mají oproti stolnímu počítači s kamerou výhodu, neboť obsahují mimo jiné i GPS, kompas a gyroskop. Díky



Obrázek 6: Virtual-Reality Continnum [9]

těmto senzorům můžeme určovat polohu kamery v prostoru a směr pohledu kamery. Virtuální realitu je navíc možno rozšířit o stažená data z Internetu.

Využití rozšířené reality je velice pestré. Od medicíny, kde lze pomocí dat z rentgenu nebo ultrazvuku vymodelovat různé nálezy v těle pacienta a lékařům tak poskytnout více informací pro zahájení léčebného plánu. Další uplatnění můžeme nalézt například ve vzdělávacích programech. Student má možnost promítnout různé geometrické struktury v trojrozměrném prostoru. Mohou si doslova scénu projít a podívat se na ni z různých pohledů. Dále se virtuální realita uplatňuje v herním průmyslu. Zesiluje estetický prožitek hráče a také ve strojírenství při údržbě přístrojů nebo složitých motorů [6].

V poslední době, kdy rozšířená realita na mobilních zařízeních zažívá svůj rozmach, společnosti vyvíjející mobilní systémy se rozhodli vyvinout nativní knihovny pro tvorbu aplikací využívající právě tuto technologii. Spolu s vydáním iOS 11 byl představen i ARkit, což je v podstatě nástroj pro rozmísťování různých objektů do náhledu kamery.

ARkit dokáže sám rozlišit vodorovné a svislé plochy a celkově rozložení objektů v zachycené scéně. Poté můžeme předměty, pohybem prstu, rozmístit různě po okolí a při otáčení kamery zůstanou na stejném místě. Existují například aplikace, jenž dovolují pohybovat s virtuálním nábytkem v místnosti nebo měřit vzdálenost pomocí gesta na obrazovce. Platforma Android v tomto technologickém směru vydala ekvivalent ARkit na chytré telefony s Android 7 a výše, nesoucí název ARcore.[11]

Obrázek 7: Ukázka mobilní aplikace Pokémon Go, která využívá rozšířenou realitu [5]

5.3 JavaScript

JavaScript je multiplatformní, objektově-orientovaný, skriptovací jazyk pro vytváření interaktivních webových aplikací. JavaScript se sám o sobě rozděluje na dva typy podle toho, na jaké straně běží. Na straně klienta je základ jazyka rozšířen o manipulací s DOM. Například různé rozšíření dovolují aplikaci reagovat na vyvolané události ze strany uživatele.[28] Na druhou stranu, je-li JavaScript na serverové straně, umožňuje funkcionality jako je připojení k databázi nebo práci se soubory na serveru.

Jako webový server se hojně využívá Node.js. Node.js je postaven na JavaScriptovém enginu V8 od společnosti Google. Tento webový server podporují i cloudové služby jako je Amazon Web Sevices nebo Microsoft Azure.

Existuje celá řada jazyků kompilujících se do JavaScriptu. Můžeme je rozdělit do dvou kategorií. Prvními jsou zcela nové jazyky, které mají odlišnou syntaxi a nelze je použít do stávajících projektů, jako je například CoffeeScript. Druhá kategorie obsahuje jazyky rozšiřující současný JavaScript. Tyto jazyky se snaží do jisté míry zachovávat dopřednou kompatibilitu. Mezi zástupce této kategorie patří kupříkladu Babel nebo TypeScript. [30]

5.3.1 Skriptovací jazyk ECMAScript

ECMA international funguje jako evropská asociace, která standardizuje JavaScript. Poslední stabilní verze tohoto jazyka bylo vydána v roce 2017 s pořadovým číslem osm. ES6 přinesl mnoho změn. Těmi nejzásadnějšími je bezpochyby přidání syntaxe pro třídy, které povolují prototypovou dědičnost, rodičovské metody, statické metody a konstruktory. Další výraznou inovací je přidání konstant a modulů, jež přehledněji rozdělují kód do několika souborů.

Výpis 1 ukazuje jednoduchý příklad implementace tříd v standardu ES6. Můžeme zde také vidět syntaxi pro dědičnosti jednotlivých tříd. Komentáře zobrazují části kódu v jiných souborech.

```
//app.js
import Animal from 'my-module.js';
class Dog extends Animal{
}

//my-module.js
export default class Animal{
}
```

Výpis 1: Demonstrace modulu v ES6

6 PhoneGap aplikace s rozšířenou realitou

Tato část práce se věnuje implementaci referenční aplikace ve frameworku PhoneGap. Pro implementaci byla využita verze 8.0.0. V frameworku PhoneGap je možné využít templates, což jsou již připravené šablony základních aplikací, rozšiřující PhoneGap o knihovnu nebo přímo o front-endový framework určený pro vývoj single-page webových aplikací. Existují vzorové šablony pro PhoneGap aplikace vytvořené na základech Vue.js, Framework7 nebo React. Všechny šablony jsou open-source a uloženy na veřejném repozitáři serveru GitHub[1].

Z důvodu představení rozšířených možností aplikace vyvíjené pomocí multiplatformního nástroje a ne tedy představení best practices vývojových technik, byla aplikace vytvořena na základní šabloně frameworku PhoneGap.

6.1 Návrh aplikace

Grafický návrh je velmi důležitou součásti aplikace, neboť špatný estetický vzhled, může uživatele odradit od jejího prvotního stažení. Kvalitně zpracovaná vizuální stránka pomůže uživateli přehledně a plynule ovládat aplikaci. Zásadní funkcí aplikace bude zobrazování informačních štítků jednotlivých hydrometeorologických stanic.

Uživateli proto musí být jako první zobrazen náhled rozšířené reality. Štítky budou mít rozličné velikosti, podle toho, je-li stanice daleko či blízko. Stanice blíž uživatele budou mít přirozeně celkovou velikost štítku větší, než ty vzdálenější. Dále musíme při implementaci brát ohled na počet zobrazených stanic, jelikož v okolí velkých měst je větší hustota měřících stanovišť. Jednotlivé štítky se mohou překrývat a stát se nečitelné. To by kazilo uživatelský dojem. Uživatel si z tohoto důvodu bude moc nastavit v jakém rozmezí vzdálenosti se budou stanice zobrazovat.

Jestliže uživatel bude mobilní zařízení držet ve vodorovné poloze se zemí bude zobrazena mapa, která uživatele informuje o jeho poloze a o poloze všech okolních stanic. Na této mapě bude zobrazena také kružnice určující vzdálenost zachycených meteostanic na úrovní rozšířené reality.

6.1.1 Návrh uživatelského rozhraní

Na obrázku 8 je možné vidět grafický návrh vzhledu aplikace. Návrh vlevo zobrazuje náhled na rozšířenou realitu a její uživatelské rozhraní. V levém horním rohu se nalézá ikona, pro zobrazení dalších informacích o aplikaci. V pravém horním rohu bude umístěn kompas, zobrazující aktuální směr, jakým se uživatel zrovna dívá.

Spodní slider slouží k ovládání rozpětí vzdáleností, mezi kterým se budou zobrazovat štítky stanic. Poté jsou na obrázku vidět dva informační štítky zobrazující aktuální hodnotu čistoty ovzduší.

[1] https://github.com/phonegap/

Je zřetelné, že štítek stanice blíže uživateli je větší než ten od vzdálenější stanice. Náhled na pravé straně ukazuje detail po kliknutí na štítek. Jsou zde vidět detailnější informace o stanici s naměřenými hodnotami jednotlivých veličin.

Obrázek 8: Návrh uživatelského rozhraní pro rozšířenou realitu

6.2 Vývoj

Tato kapitola informuje o jednotlivých krocích vývojového procesu referenční aplikace, implementující prvky rozšířené reality. Cílem této aplikace je předvést rozšířené možnosti funkcionalit mobilního telefonu při multiplatformním vývoji.

6.2.1 Rozšíření jQuery a jQuery Mobile

JQuery je knihovna rozšiřující JavaScript, o funkce pro snadnější práci s DOM. Dnes se jedná o jednu ze základních knihoven pro vývoj webových prezentací. JQuery Mobile je další rozšíření z rodiny jQuery Foundation, rozvíjející klasickou knihovnu jQuery o funkcionality specifické pro mobilní zařízení. Jedná se například o různé rozpoznávání gest a prvky uživatelského rozhraní. Tyto prvky jsou responzivní a využitelné pro všechny platformy.[14]

Další technologe, spadající pod Javascript resp. jQuery, se na nazývá Ajax. Ajax neboli Asynchronous JavaScript and XML umožňuje měnit obsah stránky bez nutnosti ji celou znovu načíst. Zejména se jedná o asynchronní zpracování požadavků plynoucích z různých webových služeb. Volání Ajax má velice jednoduchou syntaxi. Musí se samozřejmě specifikovat URL adresa, typ požadavku má být odeslán a případně parametry s daty. K požadavku je možné dále připojit další funkce vykonané při úspěchu nebo neúspěchu.

Z názvu vyplývá, že zprostředkovává pouze přenost dat XML, který byl z historického hlediska více využíván. Dnes je však více popularizován formát JSON, jenž je úspornější a jednodušší vyhledávání.

Pro nativní vzhled UI existuje knihovna OnsenUI, která zavádí speciální HTML tagy esteticky uzpůsobené pro mobilní vzhled systému Android i iOS. Tato knihovna má také portace do populárních webových frameworků jako je React, Vue nebo Angular.

```
<div data-role="rangeslider">
   <label for="range-1a">Range min</label>
      <input type="range" name="range-1a" id="range-1a" min="0" max="50" value=
         "20">
   <label for="range-1b">Range max</label>
   <input type="range" name="range-1b" id="range-1b" min="0" max="50" value="30
      ">
</div>
```

Výpis 2: Zdrojový kód HTML elementu range slider jQuery Mobile

6.2.2 Navigator

JavaScript nativně implementuje objekt *navigator*, jehož jediným úkolem je poskytovat rozhraní pro stav a identitu uživatelovy strany. Přes něj je možné ovládat také některé hardwarové prvky

zařízení. Jeho využitelnost jde nejlépe poznat na mobilních telefonech. Díky němu dokáže aplikace například ovládat vibrace telefonu, přistupovat k datům senzorů nebo pořizovat fotografické snímky a videa.

```
onSuccess = function(position){
    //do something
}

function onError(error) {
    alert('code: '    + error.code    + '\n' +
          'message: ' + error.message + '\n');
}

navigator.geolocation.getCurrentPosition(onSuccess, onError)
```

Výpis 3: Kód pro získání geolokace pomocí prohlížeče

6.2.3 Lokální uložiště

Skrze API vnitřního webové prohlížeče aplikace je možné ukládat data přímo do mobilního zařízení. K dispozici jsou dvě možnosti, jak data ukládat. První z nich se jmenuje *localStorage* a slouží pro dlouhodobé uložení dat přímo na zařízení a data poté nejsou smazána ani po ukončení aplikace. Oproti tomu *sessionStorage*, dokáže ukládat data během běhu aplikace a po ukončení jsou tyto data smazány. Oba objekty pro ukládání dat implementují tři základní funkce. Funkce setItem nastaví hodnotu dané proměnné, dále funkce getItem vrátí hodnotu proměnné a funkce removaItem zadanou proměnnou smaže.

6.2.4 Kamera

Jak již bylo zmíněno, PhoneGap patří mezi hybridní multiplatformní frameworky. Aplikace je zabalena spolu s jádrem webového prohlížeče a skrze webové technologie lze přistupovat k nativnímu API hardwaru. Přístup je však omezen na určité funkce a daná omezení závisí na platformě, na niž aplikace běží. Jako příklad můžeme uvést obraz kamery vykreslovaný do HTML elemetu canvas. Tuto funkci dovoluje pouze operační systém Android. Z bezpečnostních důvodů nemůže iOS přistupovat k některým částem hardwaru zařízení přes webový prohlížeč.

Apple blokuje volné použití kamery přes prohlížeč. Aby mohla referenční aplikace přistupovat ke kameře, je nutné využití pluginu, který skrze nativní kód dokáže ovládat kameru i na platformě iOS. Pro získání dat z kamery muselo být využito Cordova knihovny Camera Preview, umožňující vykreslení obrazu z kamery přímo do HTML elementu a dále i možnosti získání dalších informací Kamery. Pluginy se instalují jednoduchým příkazem do konzole uvedeném v příloze.

```
let options = {
  x: 0,
  y: 0,
  width: window.screen.width,
  height: window.screen.height,
  camera: CameraPreview.CAMERA_DIRECTION.BACK,
  toBack: true,
  tapPhoto: false,
  tapFocus: false,
  previewDrag: false
};

CameraPreview.startCamera(options);
```

Výpis 4: Kód pro nastavení Camera Preview

Úryvek kódu předložený výše, nastavuje základní atributy pro Camera Preview. Proměnné *x, y, width, height* určují, kde se má pohled kamery nacházet a jakou má mít velikost. Dále se musí nastavit, jaká z kamer na mobilním zařízení se má používat a zda se má element zobrazovat na pozadí. Velice důležitá je funkce getHorizontalFOV, jenž vrací horizontální zorný úhel kamery, nezbytný pro přesné zobrazování prvků rozšířené reality. Náhled kamery bohužel nedokáže reagovat na orientaci zařízení. Z tohoto důvodu je aplikace navržena pro orientaci na výšku zařízení, nebo-li portrét.

6.2.5 Data API

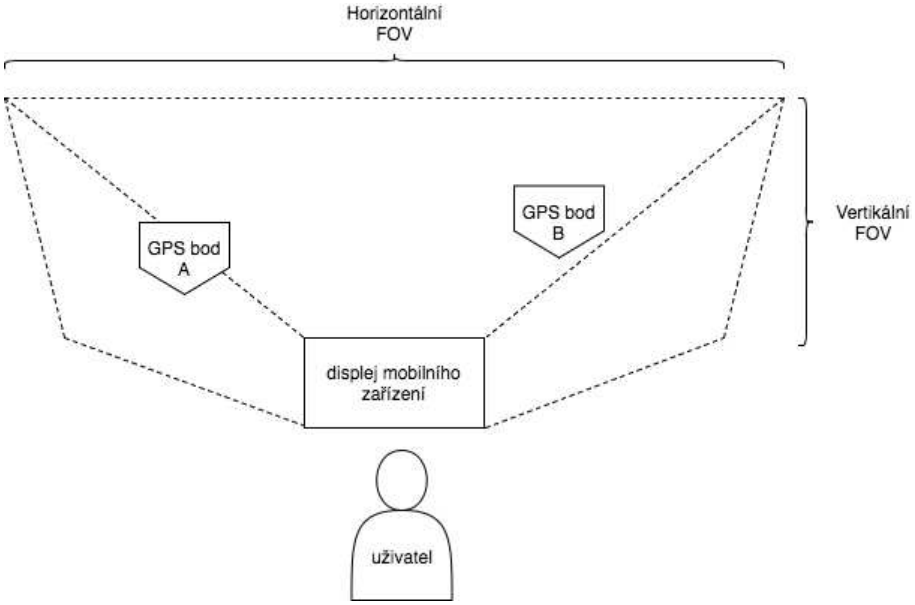
Získávání informací probíhá přes datové rozhraní Českého hydrometeorologického ústavu. Informace jsou celkem ze 135 meteorologických stanic rozmístěných po českém území do 14 krajů. Jsou zde i záznamy pocházející z některých stanic Polska a Rakouska. Interval měření se pohybuje od 1 hodiny do 24 hodin, v závislosti na dané veličině. Jestliže ve stanici není měřící zařízení pro některou z veličin, posílá se záporná hodnota, jako označení pro neměřený prvek znečištění. Data můžeme získat jak ve formátu JSON, tak i ve formátu XML.

Dále data předávají GPS souřadnice zeměpisné šířky a délky, na nichž se nachází meteorologické stanice. Pro informaci o výšce, je zapotřebí staticky implementovat slovník s unikátním kódem stanice a nadmořské výšky, kterou udává česky hydrometeorologický ústav na svých webových stránkách [2].

[2] https://goo.gl/ktqB77

6.2.6 Implementace rozšířené reality

Implementaci rozšířené reality vyžaduje data z magnetometru nebo gyroskopu, GPS lokalizaci a rozšířené funkce kamery. Aplikace v reálném čase vykresluje virtuální dvojrozměrné štítky s popisem hydrometeorologických stanic. Po kliknutí na konkrétní štítek stanice se uživateli zobrazí detailnější informace o stavu jednotlivých měřených elementů a samotné stanici. Po stažení dat z API českého hydrometeorologického ústavu, dochází setřídění stanic podle vzdálenosti. Při testování byla stanovena největší vzdálenost v okruhu 50km od pozice uživatele. Stanice dále už nemají pro uživatele tak vypovídající informační hodnotu. Jestliže má stanice menší vzdálenost než 50km, bude umístěna přímo do DOM aplikace.



Obrázek 9: Modelový nákres implementace rozšířené reality

Na obrázku výše, můžeme vidět modelové zobrazení rozšířené reality na displeji mobilního zařízení. Horizontální a Vertikální úhel je nezbytný pro stanovení hranic, mezi nimiž se musí 2D virtuální objekty zobrazovat. Pomocí magnetometru poté přijímají data o směru pohledu kamery. Některé zařízení nemají v sobě magnetický senzor a aplikace poté nemusí fungovat správně.

Pro tyto případy byla nejdříve využita technologie GPS pro počítání směru na základě přímočarého pohybu po krátké vzdálenosti, následně byl vektor vzdálenosti převeden na úhel ve směru hodinových ručiček od severu. Při testování se však vzdálenost lišila od pár metrů až desítkám metrů v závislosti na síle signálů a přesnosti zaměřené pozice.

Tento princip by mohl být pro uživatele krajně nepohodlný, proto autor práce dal přednost využití gyroskopu. Uživatel musí vědět předem, jakým směrem je server a v aplikaci se kliknutím na kompas nastaví pozice severu. Z této pozice se následně budou počítat úhly pro vykreslování objektů.

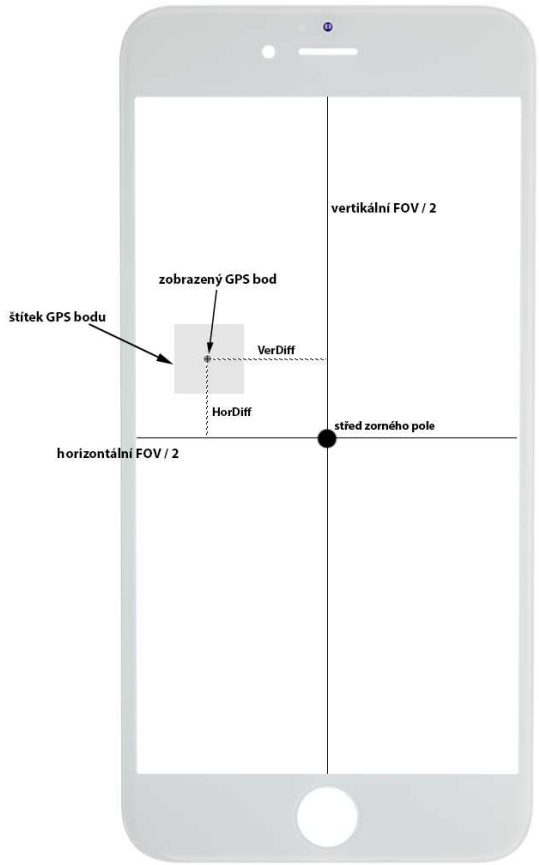
Plugin Camera Preview bohužel nedokáže vrátit hodnotu vertikálního úhlu kamery. Tato hodnota se musí dopočítat pomocí kódu uvedeném níže. Hodnota horizontálního FOV se získá zavoláním funkce getHotizontalFOV a její hodnota se vrátí ve stupních. Pro vertikální úhel se poté stupně musí převést na radiány a dále pomocí vzorce ze zdroje [33] vypočítá vertikální úhel kamery. V posledním kroku dochází ke převodu na úhlové stupně.

```
function countVerticalFOV(hFOV){
   hFOVRad = hFOV * Math.PI / 180
   var vFOVrad = 2*Math.atan(Math.tan(hFOVRad/2)*window.screen.height/window.
       screen.width);
   var vFOV = vFOVrad * (180 / Math.PI)
  return vFOV
}

var horizontalFOV = CameraPreview.getHorizontalFOV();
var verticalFOV = countVerticalFOV(horizontalFOV)

var horizontalStep = window.screen.width/horizontalFOV;
var verticalStep = window.screen.height/verticalFOV;
```

Výpis 5: Demonstrace kódu pro vypočítání vertikálního zorného pole kamery

Displej je nutné rozdělit do virtuální mřížky. Musíme uvažovat o vzdálenosti ve směru horizontálním, kterou dostaneme podílem šířky okna a horizontálního úhlu kamery a dále i vzdálenost ve vertikálním směru, jenž naopak získáme podílem výšky okna a vertikálního úhlu kamery. Tímto nám výjde vzdálenost dvou sousedících stupňů v pixelech, promítanou na displeji zařízení. vzdálenosti jsou označeny jako *horizontalStep* a *verticalStep*. To jsou hodnoty minimálního posunu jednoho 2D objektu na virtuální vrstvě.

Vzhledem k tomu, že algoritmus bude počítat pouze se souřadnicemi v určitém malém okolí od pozice uživatele, nemusí být tudíž bráno zakřivení zemského povrchu u počítání vertikálního úhlu mezi dvojicí GPS bodů. Po získání všech potřebných hodnot, algoritmus přesune všechny 2D objekty stanic do středu obrazovky. Následně je vypočítán rozdíl vertikálního a horizontálního úhlu od středu zorného úhlu uživatele a pomocí tohoto rozdílu, jsou umístěny jednotlivé štítky. Úhel může nabírat záporných a kladných hodnot, v závislosti na jaké straně od středu zorného pole se nachází.



Obrázek 10: Demonstrace umístění 2D objektů na displej

Obrázek s číslem 10 představuje princip umísťování 2D objektů na displej mobilního zařízení. Od středu zorného pole se spočítá horizontální a vertikální vzdálenost postupem, který byl již zmíněn. Vzdálenost od středu horizontálního a vertikálního úhlu je označena *HorDiff* a *VerDiff*.

Níže uvedený výpis 6 kódu představuje callback funkci, která se zavolá pokaždé, když zařízení změní svou orientaci v prostoru. Funkce je tedy zavolána při změně hodnoty magnetometru nebo gyroskopu. Nejdříve se zkontroluje, zda-li bod patří do rozmezí vzdáleností, které nastavil uživatel. V dalších kroku funkce počítá horizontální a vertikální úhel a poté jsou 2D objekty posouvány o danou vzdálenost.

Pro korekci se musí ještě posunou o polovinu své šířky a výšky, kvůli umístění do středu zobrazeného zeměpisného bodu. Posouvání se provádí přes změnu CSS atributů HTML elementu. Animace přesunu každého štítku stanice má drobné zpoždění 100ms, aby bylo dosaženo větší vizuální plynulosti a lepšího estetického dojmu.

```
window.addEventListener('deviceorientation', function (e) {
```

```
    a = Math.floor(e.alpha);
    b = Math.floor(e.beta);
    g = Math.floor(e.gamma);

    if (e.webkitCompassHeading) {
      a = e.webkitCompassHeading;
   }

   $('#ARoverlay').children('.point').each(function () {

      if($("#range-min").val() < $(this).data("distance") && $("#range-max").
          val() > $(this).data("distance")){
         $(this).css("opacity", "1")
      }else{
         $(this).css("opacity", "0")
      }

      horizontalAngleDiff = Math.floor((a - point_Vangle + 180 + 360) % 360 -
          180)
      verticalAngleDiff = Math.floor(((b-90) - elevation + 180 + 360) % 360 -
          180)

      $(this).css("left", horizontalStep*(horizontalAngleDiff)+(center-35));
      $(this).css("top", verticalStep*(verticalAngleDiff)+(center-45));

      }
  }
```

Výpis 6: Demonstrace algoritmu pro počítání pozice 2D objektů

Pro výpočet vzdálenosti bodů byla využita Haversine formule. Což je metoda jak poměrně přesně spočítat vzdálenost dvou bodů, nacházejících se na povrchu koule. Země má v obvodu okolo 40 000 kilometrů, proto její zakřivení při malých vzdálenostech může být zanedbáno. Avšak při vzdálenostech vyšších než 20 kilometrů je nezbytné, kvůli přesnějšímu výsledku, brát v potaz kulovité zakřivení zemského povrchu.[29] Níže je představen JavaScriptový kód počítající vzdušnou vzdálenost každé stanice.

```
getDistance: function (lat1, lng1, lat2, lng2) {
    var p = 0.017453292519943295; // Math.PI / 180
    var c = Math.cos;
    var a = 0.5 - c((lat2 - lat1) * p) / 2 +
```

```
        c(lat1 * p) * c(lat2 * p) *
        (1 - c((lng2 - lng1) * p)) / 2;

    return parseFloat((12742 * Math.asin(Math.sqrt(a))).toFixed(2));
   }
```

Výpis 7: Kód pro vypočítání vzdálenosti mezi dvojicí GPS souřadnic

6.2.7 Implementace náhledu mapy

Cordova GoogleMaps je doporučený plugin pro integraci náhledu geografické mapy. Tento plugin slouží ke komunikaci s nativními knihovnami Google Maps API a Google Maps SDK. Už z názvu pluginu lze vydedukovat, že byl napsán původně pro framework Apache Cordova. PhoneGap jej však může využívat také. Před samotným použití pluginu ve zdrojovém kódu je zapotřebí zadat API klíč od Google Map API, který je možné vygenerovat za pomocí přihlášení přes uživatelský účet Google. Tento klíč se následně přidá do nastavení pro sestavení aplikace. Jakmile aplikace začne pracovat s polohou uživatele, platforma Android i iOS musí uživatele požádat o použití jeho polohy. Text tohoto okna může vývojář dále upravit v souboru pro sestavení ve výpisu zdrojového kódu 8. Bohužel u operačního systému iOS plugin nepodporuje Apple Maps, na které mohou být uživatelé této platformy více zvyklí.

```
<plugin name="cordova-plugin-googlemaps" spec="^2.2.9">
    <variable name="API_KEY_FOR_ANDROID" value="(api key)" />
    <variable name="API_KEY_FOR_IOS" value="(api key)" />

    <variable name="LOCATION_WHEN_IN_USE_DESCRIPTION" value="message" />
    <variable name="LOCATION_ALWAYS_USAGE_DESCRIPTION" value="message" />
  </plugin>
```

Výpis 8: Ukázka souboru pro nastavení PhoneGap build

Výpis zdrojového kódu s číslem 9, představuje prvotní inicializaci pluginu zobrazující náhled geografické mapy aplikace. Funkce *mapInit* nejdříve určí HTML element nacházející se DOM podle unikátního ID *map-canvas*. Do tohoto elementu bude poté zobrazen náhled mapy. Následně se přidá event, jenž spustí funkci *onMapReady*. Poté, co bude plugin plně načten a připraven. V této funkci se už mohou vykonávat dílčí části, jako například načtení značek na mapě určující polohu meteorologických stanic.

```
function onMapReady() {
   Mapfill(stationsData);
}
```

```
mapInit: function {
  var div = document.getElementById("map-canvas");

  map = plugin.google.maps.Map.getMap(div);

  map.addEventListener(plugin.google.maps.event.MAP_READY, onMapReady);
}, false);
```

Výpis 9: Ukázka inicializace zobrazení mapy

6.3 Ladění

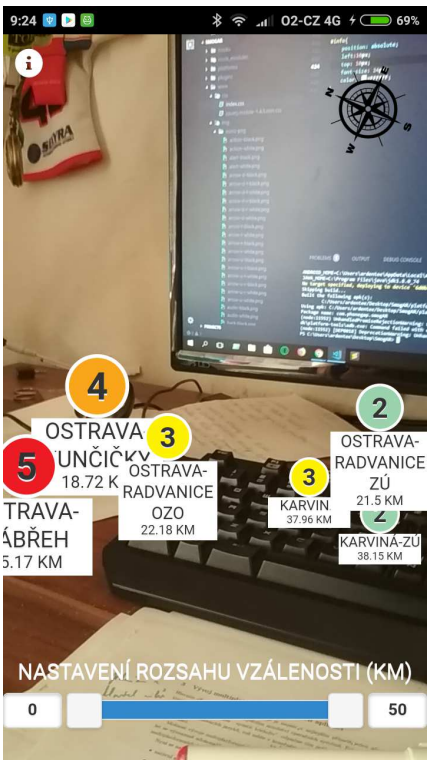
Pro ladění PhoneGap aplikace je zapotřebí dodatečné nastavení projektu. Před zahájením samotného ladění musí být přidána dodatečná nastavení. Například nastavení cest pro Android SDK.

Vzhledem k tomu, že se jedná o vývoj hybridní aplikace, mohou se nativní části ladit přímo pomocí Android Studia nebo vývojového prostředí Xcode, v závislosti na platformě. V takovém případě se, při hledání chyb aplikace, smí používat breakpointy a výpisy do konzole. Obdobně lze ladit nativní části v Android Studiu, což je vývojové prostředí pro vytváření nativních aplikací pro systém Android.

Protože PhoneGap využívá ke svému běhu Webview, lze využít i Safari Web Inspektor a ladit tak část aplikace, napsanou v JavaScriptu. V nastavení však musí být povolena funkce vzdáleného ladění. Alternativa pro Safari Web Inspektor u systému Android je Chrome Remote Debugger, jenž zastupuje stejnou funkci.

6.4 Výsledná aplikace

Na obrázku 11 je vyobrazena rozšířená realita SmogAR. Aplikace byla spuštěna ve městě Studénka, při pohledu ve směru na město Ostrava.



Obrázek 11: Obrázek aplikace SmogAR s rozšířenou realitou.

6.5 Silné stránky

Na podobném principu jako je Live reload u React Native pracuje i funkce Hot module replacement, která vkládá upravené části kódu do spuštěné aplikace, aniž by musela být opakovaně sestavena. Tímto se vývoj a zejména tvorba a ladění uživatelského rozhraní znatelně urychlí.

Pokud je aplikace vyvíjena programátorem, specializujícím se na webové aplikace nebo webové stránky, nemusí být prostředím PhoneGap nikterak zaskočen, protože většina principů je velice podobných s tvorbou klasické webové aplikace.

Další z věcí, v nichž PhoneGap vyniká, je obrovská databáze různých rozšíření třetích stran. Buď jsou to přímo knihovny vytvořené autory PhoneGap nebo komunitou. Pro ovládání některých specifických hardwarových prvků jsou tato rozšíření nezbytná. Znovupoužití vytvořených pluginů šetří celkový čas vývoje.

PhoneGap jako multiplatformní vývojový nástroj, téměř doslova splňuje paradigma "write once - run anywhere". Všechny platformy jsou založeny na stejném zdrojovém kódu pro logiku i uživatelské rozhraní aplikace.

Mezi další důležité vlastnosti patří i to, že není závislý na konkrétním front-endovém frameworku, jako například React Native na React nebo Ionic na Angular. Tímto má vývojář určitou možnost volby a výběr technologií je více flexibilní.

6.6 Slabé stránky

Kvůli rozhraní webového prohlížeče uvnitř aplikace je nejvýraznější slabinou PhoneGap výkon. U náročnější funkcionalit, může tento fakt způsobit znatelné zpomalení. Uživatelské rozhraní a animace nebudou dosahovat takové plynulosti, jako je tomu u nativních aplikací. Nevýhodou také může být absence některých základní vývojových rozšířeních, při založení nového projektu. Například se může jednat o předem nastylované HTML elementy, jenž budou do jisté míry simulovat nativní vzhled.

7 Referenční aplikace pro srovnání React Native a PhoneGap

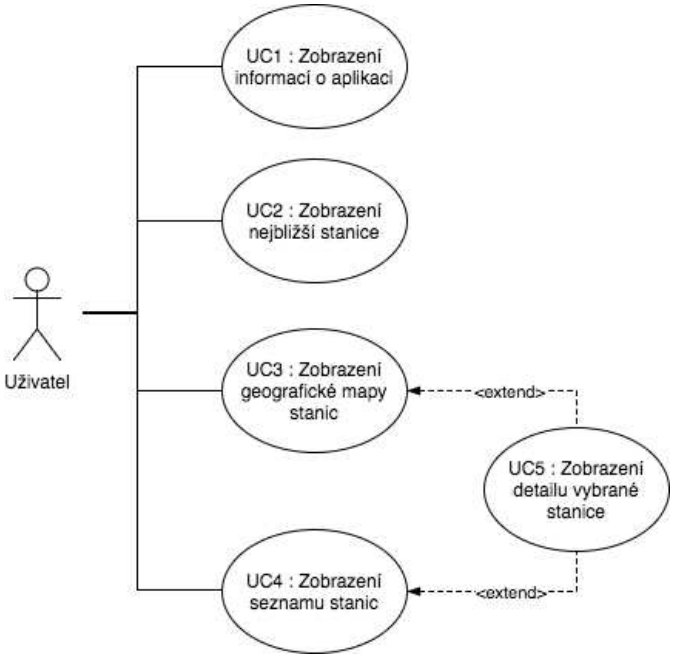
V rámci práce byla vytvořena aplikace, ve dvou různých multiplatformních nástrojích. Tato aplikace slouží k porovnání možností vývoje v React Native a PhoneGap. Srovnávací aplikace vytvořená ve PhoneGap bude vycházet z aplikace SmogAR. Aplikace bude však bez implementace rozšířené reality a dále bude obměněna o nové datové API a některé nové prvky uživatelského rozhraní.

7.1 PhoneGap aplikace pro porovnání frameworků

Tato aplikace bude vycházet z aplikace SmogAR. Vzhledem k tomu, že většina vývojových postupů již byla uvedena u aplikace s rozšířenou realitou, nebude rozepsán širší postup pro implementaci srovnávací aplikace v frameworku PhoneGap.

7.2 Návrh aplikace

První část návrhu tvoří diagram případů užití. Aplikaci tvoří 5 případů označených UC1 až UC5. Primární cíl aplikace bude zobrazovat uživateli data z meteostanic. Po načtení aplikace mu tedy budou zobrazena nejzajímavější informace, kterou je detail nejbližší stanice s naměřenými hodnotami. Uživatel poté bude mít možnost procházet seznamem všech stanic, jenž budou seřazeny podle vzdálenosti od uživatele a také na mapu stanice. Náhled mapy bude zobrazovat stanice s naměřenou hodnotou srážek nebo teploty.



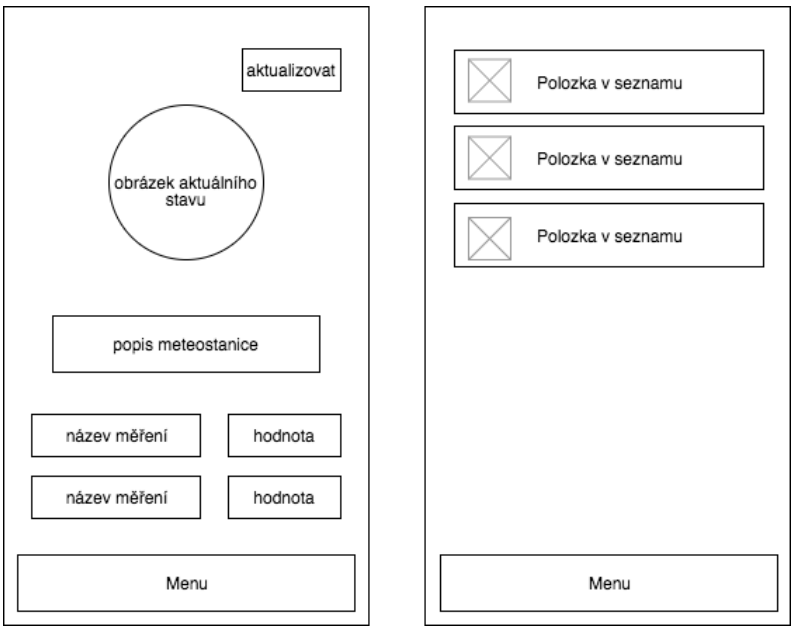
Obrázek 12: Rozvržení jednotlivých Views aplikace

8 React Native aplikace

Tato část popisuje vytvoření aplikace, pro porovnání vývoje ve frameworku React Native a PhoneGap. Aplikace byla pracovně pojmenována jako Air. Základem aplikace bude datový zdroj meteorologických stanic, o jejichž datech bude aplikace informovat uživatele. Aplikace musí zobrazovat výpis stanic s možností detailnějšího náhledu. Dále poskytuje uživateli geografickou mapu s GPS souřadnicemi meteostanic. Aplikace bude využívat GPS lokalizace pro zobrazení polohy uživatele a vzdáleností měřících stanic. V této části práce je popsán vývojový proces při vytváření aplikace Air ve frameworku React Native verze 0.55.2.

8.0.1 Návrh uživatelského rozhraní

Uživatelské rozhraní bude založeno na vytvořeném wireframe zachyceném na obrázku 13.



Obrázek 13: Wireframe uživatelského rozhraní

8.1 Vývoj

Tato část práce popisuje vývoj aplikace Air v prostředí React Native. Jsou vylíčeny jednotlivé stěžejní části aplikace a základní principy frameworku.

8.1.1 Komponenta

Základním stavebním prvkem React Native je komponenta. Hlavní myšlenka je založena na tom, že každé uživatelské rozhraní se skládá z komponent, do nichž proudí data a renderují nějaký

prvek UI. Komponenta může ukládat svá vnitřní data do stavů, které se zapisují jako *state*. Každá komponenta má životní cyklus, skládající se z několika částí.

První částí je inicializace samotné komponenty. V této části dochází k přiřazení výchozích hodnot stavů. V druhém kroku se přiřadí výchozí hodnoty *props*. Do *props* například vstupují data z rodičovské komponenty nebo z jiného datového zdroje.

Další fáze se jmenují componentWillMount, render a componentDidMount. ComponentWillMount je metoda třídy *React.Component*, která se vykonává před samotným vykreslením komponenty, a tudíž nelze v ní žádný HTML element překreslovat. Funkce render vrací již samotné prvky DOM, které se mají vykreslit spolu s daty. Případě, že komponenta nevykresluje žádný prvek UI, může být navrácena boolovská hodnota. V Poslední fázi se provádí componentDidMount. Tato metoda se vykonává až po ukončení metody render a slouží zejména k dodatečné úpravě DOM. [34]

Uvedený zdrojový kód 10 je na první pohled podobný výpisu 3. Taktéž využívá objektu *navigator* pro práci s API hardwaru zařízení jako je tomu u frameworku PhoneGap. Tímto způsobem lze získat polohu i u běžných webových aplikací

Vzhledem k tomu, že *getCurrentPosition* je asynchronní funkce, měla by mít stav, který nabývá před vykonání metody hodnoty *null*, a poté se do stavu *position* načte vrácená pozice zařízení. Tento stav bude poté sloužit jako ukazatel, zda-li se funkce již dokončila.

Funkce s názvem *componentDidMount* je nativní metoda frameworku React Native, ukončující životní cyklus celé komponenty a volá se až po samotném vyrendrování.

Tímto se docílí, že komponenta bude renderovat prázdný prvek uživatelského rozhraní a až po ukončení se vyrendruje prvek s daty. Během zpracování se poté, místo prázdného prvku UI, může vykreslit například identifikátor aktivity, ve formě animovaného kruhu, aby byl uživatel obeznámen o prováděném procesu aplikace.

```
export default class App extends React.Component {

  state = {
    position: null
  }

  componentDidMount(){
    navigator.geolocation.getCurrentPosition(
        position => {
        console.log(position.coords.latitude);
        console.log(position.coords.longitude);
        this.setState({ position });
      },
      error => this.setState({ error: error.message }),
```

```
    { enableHighAccuracy: true, timeout: 10000, maximumAge: 1000 }
  );
}
```

Výpis 10: Kód pro získání polohy zařízení v React Native

8.1.2 Navigace a Gesta

Zpracování gest je přímo součástí View komponenty v React Native. Jedná se o zachytávání gest jako eventu, který je poté poslán do *props* komponenty View. Následně se určí metody, jaké budou volány, jakmile se spustí event. Díky tomuto systému se mohou vytvářet i komplikovaná gesta. Pro účely referenční aplikace, byla přidána knihovna react-native-navigation, jenž zprostředkovává rozhraní umožňující vytvářet menu a celkovou navigaci aplikace.

Plugin react-native-navigation je doporučen použít přímo v dokumentaci React Native. Na Výpisu 11 můžeme vidět objekt *Tabs*, používaný komponentou *TabNavigator*. Jak můžeme pozorovat, komponenta dovoluje jednoduché nastavení menu, se všemi potřebnými položkami. Každá položka menu se skládá z atributu *screen*, což je v podstatě odkaz na komponentu, která se po kliknutím na položku vyrendruje. Poté se zde nachází nastavení pro název položky a případnou zastupující ikonu. Velice užitečnou funkcí je dodatečný atribut *swipeEnabled*, přidávající gesta swipe pro pohyb po jednotlivých obrazovkách.

```
import { TabNavigator } from "react-navigation";

import InformationScreen from "../Screens/InformationScreen";
import MapScreen from "../Screens/MapScreen";
import NearestStationScreen from "../Screens/NearestStationScreen";
import StationListScreen from "../Screens/StationListScreen";

import React from 'react';
import { Icon } from 'native-base';

export const Tabs = TabNavigator({
        NearestStation: {
          screen: NearestStationScreen,
          navigationOptions: {
            tabBarLabel: 'Nejblizsi stanice',
            tabBarIcon: <Icon name="ios-pin" />
          }
        },
        Mapa: {
```

```
          screen: MapScreen,
          navigationOptions: {
            tabBarLabel: "Mapa",
            tabBarIcon: <Icon name="map" />
          }
        },
...
      {
          tabBarPosition: 'bottom',
          swipeEnabled: true
       }
```

Výpis 11: Demonstrace kódu implementuící Tab navigaci v React Native

8.1.3 Native Base

Jedná se o open-source knihovnu, implementující základní prvky uživatelské rozhraní na platformně Android a iOS. Pomocí této knihovny lze některé elementy UI sdílet mezi operačními systémy a zrychlit proces vývoje. Aplikace za použití Native Base mají stejné prvky uživatelského rozhraní jako nativní aplikace. Tímto lze dosáhnout lepší uživatelské přívětivosti a rychlejšího vývoje grafického rozhraní.

8.1.4 Data API

Aplikace pracuje s veřejným datovým API společnosti Netatmo, specialuzující se na výrobu produktů pro chytré domácnosti. Společnost poskytuje záznamy měření z osobních meteostanic, které prodávají. Po zaregistrování a získání unikátních uživatelských údajů, je možné mít, do jisté míry zdarma omezený přístup k datům jednotlivých stanic. Netatmo service vyžaduje pouze OAuth 2.0 pro autorizaci. Netatmo zpřístupňuje uvedenou url níže, ve které musí být zadáno ještě 5 povinných parametrů. Jedním z nich je unikátní klíč, získaný po registraci. Dále to jsou dvě dvojice GPS souřadnic, jenž zastupují severovýchodní a jihozápadní rohové body čtyřúhelníku. Data jsou poslána pouze ze stanic, nacházejících se v tomto čtyřúhelníku. Jedna stanice může měřit hned několik veličin, v závislosti na modulech, které jsou k ní připojeny. Kdyby stanice měla k dispozici všechny moduly, je možné získat data například o teplotě vzduchu, vlhkosti, tlaku vzduchu nebo srážkách.

Pro komunikace s Netatmo Api byl naimplementován objekt *NetatmoAPI*, obsahující metody pro volání požadavků. Jsou v něm uloženy i data pro autorizace aplikace, pro získání přístupového tokenu. Dále je zde naimplementovaná metoda *getPublicData*, která díky tokenu vrací seznam všech měřících jednotek v určené oblasti. Hraniční body oblasti se vyměří pomocí ode-

čtení nebo přičtení konstant k GPS souřadnicím uživatele. Aplikace nejdříve musí zjistit polohu uživatele a získat token, což jsou obojí asynchronní funkce.

Využívá se objektu Promise, reprezentující dokončení a vrácenou hodnotu asynchronní operace. Tento objekt byl představen se standardem ES6. Aplikace tudíž nejdříve čeká, až získá token a dokončí se geolokace. Následně se zavolá funkce vracející pole jednotlivých stanic ve vytyčené oblasti.

```
const TokenURL = "https://api.netatmo.com/oauth2/token"

export const NetatmoAPI = {
    getToken: function() {
        return fetch(TokenURL, {
          method: "POST",
          headers: {
            Accept: "application/json",
            "Content-Type":
              "application/x-www-form-urlencoded"
          },
          body: creditals
        })
          .then(response => response.json())
          .then(responseJson => {
            console.log(responseJson);
            return responseJson;
          })
          .catch(error => {
            console.error(error);
          });
    },
```

Výpis 12: Ukázka metody pro získání bezpečnostního tokenu

8.2 Implementace náhledu mapy

Dokumentace k frameworku uvádí, že náhled mapy je převzat od pluginu react-native-maps. Dříve se tuto knihovnu starala společnost Airbnb, ale později byly zdrojového kódy předány komunitě. Instalace pluginu není tak jednoduchá a nelze ji dělat podle automatického linkování jak je tomu uvedeno v příloze.

Knihovna se neustále vyvíjí a pro verzi React Native je nutná manuální instalace přes CocoaPods a Gradle. Dále je potřeba nastavit ještě další atributy jako API klíč pro Google Maps nebo zda-li používat nativní mapy na zařízeních od firmy Apple.

Hlavní komponenta react-native-maps se nazývá *MapView*, jenž zastupuje přímo náhled geografické mapu.

Pro svou inicializaci komponenty musí být zadány počáteční GPS souřadnice. V této komponentě se nachází pole markerů, což jsou podstatě značky umístěné na mapě. Na výpisu kódu 13 jsou značky přidány pomocí funkce *map*. Pro každou stanici se vytvoří jedna značka, díky které může uživatel, po kliknutí, přejít na detailnější informace. Barva jednotlivých značek je odvozena od vybraných odstínů oranžová, v závislosti na naměřené teplotě.

```
<MapView style={ styles.map }
      showsUserLocation={true} initialRegion={{
      latitude: this.props.latitude,
      longitude: this.props.longitude,
      latitudeDelta: 0.5,
   longitudeDelta: 0.5 }}>
     {this.props.screenProps.stations.map(marker => (
        <Marker
          coordinate={{
            longitude: marker.place.location[0],
            latitude: marker.place.location[1]
          }}
          pinColor={marker.temperature_color}
          onPress={() => this.navigateToDetail(marker)}
          identifier={marker.id}
        />
       ))}
     </MapView>
```

Výpis 13: Demonstrace komponenty MapView

8.3 Ladění

Ladící proces probíhá podobně jako při vývoji aplikace PhoneGap. Nativní části mohou být odladěny ve vývojářských prostředích Xcode nebo Android Studio.

React native dává možnost ladit aplikace vzdáleně přes vývojový nástroj prohlížeče Chrome. Balíčkovací server React Native spustí na lokální adrese s portem 8081 rozhraní pro výstup JavaScriptových výjimek. Přes Chrome DevTools je pak možné nahlédnout do logu aplikace.

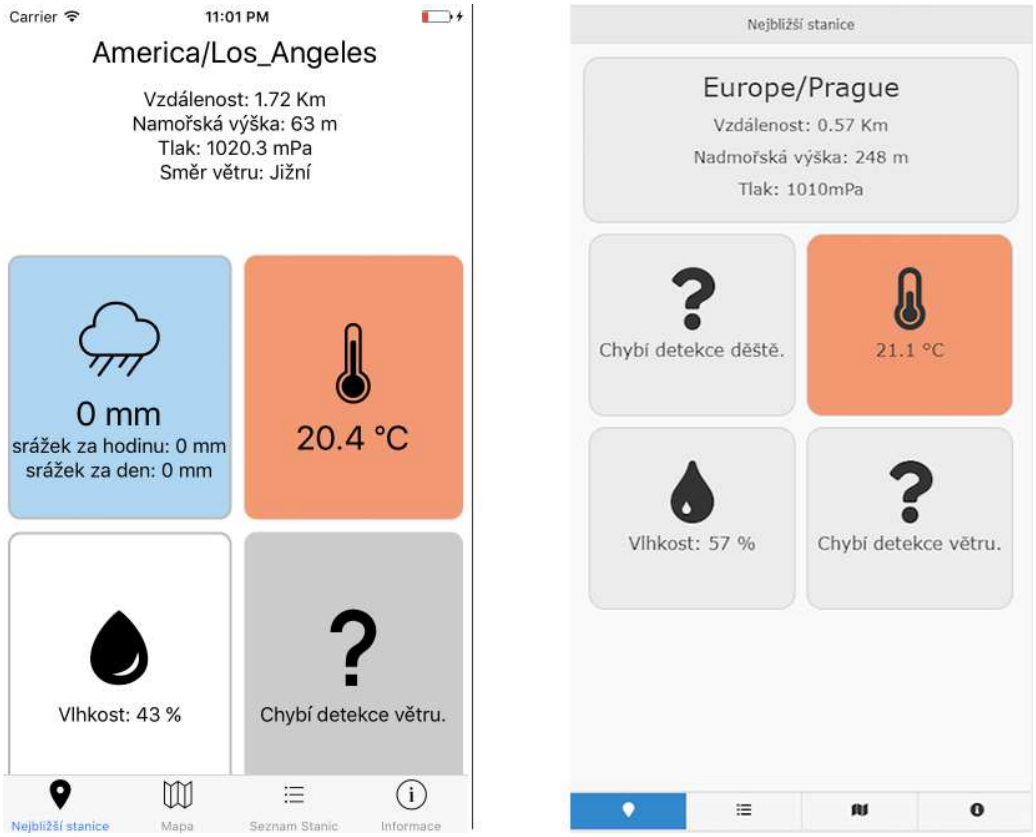
8.3.1 Rozšíření Expo

Expo je pomocná sada nástrojů a knihoven pro vývoj, sestavování a ladění projektu React Native. Princip je opět podobný jako u PhoneGap. Pro použití je nutné stáhnout mobilní aplikaci Expo client přímo do testovaného zařízení. Tato sada nástrojů není distribuována přímo s frameworkem React Native, ale musí být přidána zvlášť do projektu.

Rozšíření React Native v některých případech nepočítají s použitím Expo a jejich dodatečné nastavení pro tento nástroj poté nemusí být vždy specifikováno v dokumentaci. Dále je nezbytné nainstalovat Expo XDE nebo Expo CLI na počítač, na kterém se vyvíjí aplikace v React Native. Načež se přes uživatelský účet Expo propojí spuštěný vývojový server počítače s mobilní aplikací. Uživatelský účet slouží také pro správu a nastavení projektů Expo. Tato sada nástrojů poskytuje i vlastní SDK, jenž může přistupovat k hardwaru mobilního zařízení a proto sestavení aplikace v Xcode nebo Android Studiu není v některých případech potřeba.

8.4 Výsledná aplikace

Na obrázku 14 je finální verze aplikace Air. Levá obrazovka patří implementaci v React Native a pravá patří PhoneGap. Na obrázku se nachází úvodní náhledy obou aplikací, jenž se uživateli zobrazí po spuštění.



Obrázek 14: Ukázka aplikace Air v React Native a Phonegap

8.5 Silné stránky

Mezi výhody React Native bezesporu patří rychlost vývoje aplikace. Díky již zmíněným funkcím Hot reload a Live reload, nemusí být aplikace pokaždé instalována, jako je tomu u nativního vývoje.

Další významnou pozitivní vlastností frameworku je aktivní a rozsáhlá komunita. Firma Facebook řídí celý vývoj React Native a stará se o opravu chyb a vydávání nových verzí. Nástroj si oblíbili i vývojáři z dalších velkých společností jako je například Airbnb nebo Tesla. Rozsáhlá podpora má pozitivní dopad na vývoj a opravu chyb.

8.6 Slabé stránky

React Native je poměrně nový nástroj uvedený v lednu roku 2015. Nové verze vycházejí až několikrát do měsíce. To je známkou aktivní komunity, ale taky prudkého rozvoje a opravování chyb. Postup vývoje je tedy založen na vytvoření nativních částí pro určité specifické funkcionality. Každá nativní část je obalena v komponentě React Native a z nich je později postavena celá aplikace. Z tohoto důvodu je zapotřebí pro vývoj velkého projektu několik týmů, které se starají o nativní komponenty každé platformy.

Frekvence vydávání nových verzí nesvědčí zejména vývoji pluginů třetích stran, jenž velice často nejsou kompatibilní s předchozí verzí frameworku. Tudíž se ani moc nevyplatí psát plugin pro nějakou speciální funkci, protože poté nemusí správně pracovat s novou verzí React Native.

Velké firmy jako je například Airbnb, Tesla nebo Facebook spravují několik rozšíření React Native. Pod záštitou takových společností je do velké míry zaručena kompatibilita s novějšími verzemi frameworku. Takové firmy mají dostatek zdrojů, a je pro jejich mobilní aplikace stěžejní udržovat knihovny aktualizované. Ovšem takových příkladů není mnoho, a z tohoto důvodu pokrytí některých funkcionalit chybí.

9 Zhodnocení

Hlavní myšlenkou této práce je porovnání dvou zásadně odlišných technologií pro vývoj multiplatformních aplikací. Za tímto účelem byla vyvinuta aplikace Air v frameworku React Native a PhoneGap. Vytvořené aplikace byly otestovány na zařízeních Xiaomi Redmi 4x s Android 7.1.2. a iPhone 5 s iOS 10.3.3. Výsledné porovnání se zaměřovalo na podporu API hardwaru mobilního zařízení, funkce UI a programátorskou přivětivost. Z vybraných funkcionalit byla vybrána pouze malá část pro obecnou mobilní aplikaci. Jednotlivé soubory vlastností se budou hodnotit pomocí stupnice 1 až 4 body:

- 1 bod - chybí nebo nedostačuje
- 2 body - implementovány pouze základní funkce
- 3 body - implementováno s omezeními
- 4 body - není žádné zásadní omezení

Tabulka 2: Hodnocení různých podporovaných funkcionalit mobilního zařízení

Název Vlastnosti	PhoneGap	React Native
Kamera	3	2
Senzory	4	2
Mapa	3	4
Geolokace	4	4
Nativní knihovny pro AR	3	2
Lokální uložiště	4	4
Celkové hodnocení	22	18

Tabulka 3: Hodnocení vlastností uživatelského rozhraní

Název vlastnosti	PhoneGap	React Native
Gesta	4	4
Animace	4	4
Nativní vzhled	2	4
Celkové hodnocení	10	12

Tabulka 4 porovnává jednotlivé funkcionality PhoneGap a React Native. Jednotlivé výsledky jsou převzaty z dokumentací obou frameworků.

Tabulka 4: Porovnání funkcionalit z hlediska podpory každého frameworku

Funkcionalita	PhoneGap	React Native
Kamera	plugin	plugin
Senzory	nativně	plugin
Bluetooth	plugin	plugin
Mapa	plugin	plugin
Geolokace	nativně	nativně
Nativní AR knihovna	plugin	plugin
Vibrace	nativně	nativně
Notifikace	nativně	nativně
Přístup ke kontaktům	nativně	nativně
Grafika (2D a 3D)	nativně	plugin
Klávesnice	nativně	nativně

Ve srovnání podporovaných funkcionalit v tabulce 2 byl lepší PhoneGap a to hlavně díky zachování funkčnosti a rozsáhlé databázi pluginů. To můžeme přičíst jeho delší působnosti na poli multiplatformních nástrojů, jenž si za svou dobu vybudoval velkou komunitu vývojářů. Co se týče uživatelského rozhraní v tabulce 3, bylo jedním z kriterií nativní vzhled. Nativní vzhled je důležitý pro použití objektů uživatelského rozhraní, na které je uživatel mobilního telefonu zvyklý. PhoneGap se nesnaží o podobnost UI s nativní aplikací, ale spíše o to, aby všechny platformy měly stejné uživatelské rozhraní, nejlépe napsané jednotným zdrojovým kódem.

Další porovnání vlastností obou nástrojů přineslo testování některých rysů aplikace Air, uvedených v tabulkách. 5 a 6. Pro testování aplikace bylo využito virtuálních simulátorů obou operačních systémů. Konkrétně se jedná o simulátory mobilních zařízeních iPhone 7 a Pixel 2. Každá vlastnost náleží do sestavení určeného pro produkci, nebo-li release. Při testování byly využity profilovací nástroje, jenž jsou součástí Android Studia a prostředí Xcode. Doba sestavení byla určena z konzolového výstupu každého frameworku.

Pro měření doby startu aplikace byl použit studený start. Studený start znamená, že aplikace před svým spuštěním neměla předtím spuštěnou žádnou svou instanci na zařízení. Měření času sestavení bylo provedeno i se sestavením knihoven třetích stran, která byly využity během implementace. Doba startu aplikace znamená časový úsek od samotného startu až po kompletní inicializaci první hlavní stránky nebo komponenty. Rychlost sestavení a doba spuštění jsou uvedeny jako průměr hodnot z provedených pěti pokusů.

Tabulka 5: Porovnání vlastností aplikace Air na platformě iOS

vlastnosti	PhoneGap	React Native
Velikost aplikace	14,5 MB	23,5 MB
Doba sestavení	13,8 s	32 s
Doba startu	1,28 s	0,59 s

Tabulka 6: Porovnání vlastností aplikace Air na platformě Android

vlastnosti	PhoneGap	React Native
Velikost aplikace	12,9 MB	32,9 MB
Doba sestavení	13.2 s	38,2 s
Doba startu	1,12 s	0,65 s

9.1 Programátorská přívětivost

Programátorská přívětivost může být hodnocena velice subjektivně. Autor se bude snažit hodnotit oba frameworky objektivně, vzhledem k jeho dosavadním zkušenostem a znalostem v programování a vývoji nativních mobilních aplikací.

Pro programátora se zkušenostmi s React bude snazší naučit se pracovat v React Native, protože jsou si tyto technologie při vývoji velice podobné a většina principů zůstává stejných. Oproti tomu PhoneGap dokáže oslovit širší skupinu vývojářů, hlavně kvůli volnosti ve výběru front-endového frameworku. PhoneGap je nástroj pro vytvoření aplikací, které se nezaměřují na nativní vzhled a chování, ale spíše na funkčnost a rychlost vývoje. PhoneGap bych doporučil webovým vývojářům, kteří chtějí snadno vytvářet mobilní aplikace bez nutnosti učení se novým technologiím.

9.1.1 React Native

React native je založen na velice populárním a poměrně novém frameworku React. Celá technologie React se opírá o paradigma "learn once use anywhere", což by se ve volném překladu znamenalo "nauč se jednou, používej všude". Kde nejde ani tak o to, jestli se napíše jeden kód a ten bude spustitelný na všech platformách, ale cílem je aplikování vývojových principů React na ostatní platformy.

Pro programátora se zkušenostmi v React je jednodušší naučit se pracovat s tímto frameworkem. Principy obou jsou velice podobné, ale jsou zde některé specifické funkce. Jako příklad se může uvést stylování objektů uživatelského rozhraní. V React Native se nejedná o čistě webové elementy a proto se s nimi musí zacházet jiným způsobem.

Velmi negativní vlastností je bouřlivý vývoj a neaktuální pluginy, které ve významném počtu případů nefungují. To může být velice náročné pro udržení velkých aplikací, které provádějí zvláštní práci s API hardwaru mobilního zařízení nebo jsou závislé na spoustě knihoven třetích stran. V nově vydaných verzích frameworku React Native nemusí fungovat a přepsaní nativního kódu nebo vytvoření vlastní implementace může být pro vývoj časově a finančně náročné. Velké firmy používající React Native, spravují ve většině případů i vlastní pluginy svých aplikací. Pluginy jsou poté velice často dostupné pod nějakým typem open-source licence.

Jako příklad je možné uvést společnost Airbnb spravující několik desítek repositářů na serveru GitHub. Z velké části se jedná o rozšíření soustředěné okolo uživatelského rozhraní React Native.Tímto se zaručuje určitá kompatibilita mezi verzemi a zachování funkčnosti jejich mobilní aplikace. Pokud tedy programátor chce vyloženě vytvořit specifickou funkci, musí bezvýhradně sáhnout do nativního kódu pro obě platformy a vytvořit si vlastní rozšíření. Z tohoto pohledu je nezbytná znalost webových i nativních technologií pro vývoj aplikace.

9.1.2 PhoneGap

PhoneGap působí na poli multiplatformních nástrojů již delší dobu. To se významně podepisuje na podpoře spousty pluginů, které pracují s různými hardwarovými prvky mobilního zařízení. Rozšíření nabízená v tomto frameworku pokrývají širokou škálu funkcionalit.

Bohužel kvůli integrovanému prohlížeči uvnitř aplikace není možné dosáhnout takového výkonu jako u nativních aplikací. PhoneGap může používat široké spektrum webových vývojářů, protože není závislý na nějakém dalším frameworku, jako například React Native nebo Ionic. Je třeba ale podotknout fakt, že dnešní webové prohlížeče dovolují poměrně rozsáhlou manipulaci s hardwarem zařízení, a proto zůstává otázkou, zda-li není lepší některé aplikace vyvíjet jako čistě webové.

10 Závěr

Zásadní myšlenkou této práce je seznámit čtenáře s možnými přístupy multiplatformního vývoje mobilní aplikace. V rámci představení rozšířených funkcí byla vytvořena aplikace SmogAR. SmogAR demonstruje schopnost, vytvořit aplikaci v multiplatformním frameworku, založenou na technologii rozšířené reality. Rozšířená realita byla vybrána z důvodu přímého demonstrování několika funkcionalit hardwaru mobilního zařízení. Pro implementaci byl vybrán nástroj PhoneGap, který má ustálený vývoj a podporuje všechny potřebné funkcionality do takové míry, aby bylo možné aplikaci vytvořit. React Native nemá v tomto případě dostatečné rozšíření a zatím není pro rozšířenou realitu zcela připraven.

Některé pluginy, které pracují s API hardwaru, nefungují na novějších verzích nebo nezahrnují potřebné funkce. V React Native začínají vznikat projekty[3] snažící se o integraci ARKit a ARCore, což jsou nativní knihovny pro tvorbu rozšířené reality na mobilních operačních systémem Android a iOS. Ovšem tyto projekty jsou stále pod aktivním vývojem a mají pouze základní funkce.

Pro porovnání dvou multiplatformních nástrojů byla vytvořena aplikace s názvem Air. Pro srovnání byl vybrán PhoneGap a React Native. Oba tyto frameworky umožňují vývoj mobilní aplikace pro Android i iOS a jsou založeny na webových technologiích. Každý z nich zastupuje rozdílný přístup vývoje multiplatformní aplikace.

React Native dokáže zatím implementovat menší množství pokročilých funkcionalit hardwaru zařízení, ale vývoj tohoto frameworku jde promptně dopředu a časem se očekává, že tohle omezení zmizí. Díky použití nativních komponent je dodatečná úprava uživatelského rozhraní minimální. Existuje také několik projektů, které se zaměřují na celistvý přirozený vzhled UI platformy iOS a Android. Nejvýznamnějšími zástupci jsou open-source projekty Native Base a Shoutem UI. U React Native se předpokládá, že se vývoj ustálí a problémy s kompatibilitou verzí nebudou tak výrazné. React Native může být velice vhodný nástroj pro vývoj malých až středních aplikací, jenž nevyužívají specifické operace nad hardwarem zařízení.

Kvůli své rozsáhlé databázi pluginů a velice rychlému vývoji aplikace se stává nejenom silným nástrojem pro vývoj multiplatformních aplikací, ale je možné ho využít jako prototypovým nástrojem. Pro společnost zaměřenou spíše na vývoj webových aplikací může PhoneGap znamenat rozšíření pole působnosti o mobilní aplikace a to bez vynaložení větších prostředků na integraci této technologie do vývojového procesu.

[3] https://github.com/react-native-ar/react-native-arkit

Literatura

[1] Picking A Mobile Technology Stack [online]. Santosh Shinde, [cit. 2018-02-20]. Dostupné na: http://santoshshinde2012.blogspot.cz/2016/02/picking-mobile-technology-stack.html

[2] Smartphone OS Market Share, 2017 Q1 [online]. 2018 IDC, [cit. 2018-02-21]. Dostupné na: https://www.idc.com/promo/smartphone-market-share/os

[3] Mobile web usage overtakes desktop for first time [online]. James Titcomb , [cit. 2018-02-21]. Dostupné na: https://www.telegraph.co.uk/technology/2016/11/01/mobile-web-usage-overtakes-desktop-for-first-time/

[4] Mobile operating system. Wikipedia: the free encyclopedia [online]. San Francisco (CA): Wikimedia Foundation, 2001- [cit. 2017-03-11]. Dostupné na :https://goo.gl/iQ8WEU

[5] Pokémon Go Will Make You Crave Augmented Reality [online], Om Malik, [cit. 2018-02-23]. Dostupné na: https://www.newyorker.com/tech/elements/pokemon-go-will-make-you-crave-augmented-reality

[6] AUGMENTED REALITY THEORY AND APPLICATIONS [online], Ajunewanis Ismail, Zakiah Noh 2016, [cit. 2018-02-23]. Dostupné na: https://goo.gl/Wika31

[7] SCHMALSTIEG, D. a Tobias HÖLLERER. Augmented reality: principles and practice. Boston: Addison-Wesley, 2016. Addison-Wesley usability and HCI series. ISBN 978-0-321-88357-5.

[8] Virtual Reality vs. Augmented Reality [online], Augment 2015, [cit. 2018-02-23]. Dostupné na: http://www.augment.com/blog/virtual-reality-vs-augmented-reality/

[9] Augmented Reality: A class of displays on the reality-virtuality continuum [online], Paul Milgram, Haruo Takemura, Akira Utsumi, Fumio Kishino [cit. 2018-02-23]. Dostupné na: https://goo.gl/JN6kUi

[10] MOBILE APP DEVELOPMENT: NATIVE OR WEB? [online], Adrian Holzer, Jan Ondrus [cit. 2018-03-15]. Dostupné na: http://www.janondrus.com/wp-content/uploads/2008/05/WEB2012.pdf

[11] ARKit [online]. 2018 Apple Inc., [cit. 2018-04-04]. Dostupné na: https://developer.apple.com/documentation/arkit/

[12] HERMES, Dan. Xamarin Mobile Application Development: Cross-Platform C# and Xamarin.Forms Fundamentals. California: Apress, 2015. ISBN 978-1-4842-0214-2.

[13] State of Art Approaches to Build Cross Platform Mobile Application [online]. Bhushan S. Thakare and Dhanashree Shirodkar and Naghma Parween and Shama Parween and Spyros

Xanthopoulos and Rahul C. P. Raj and Seshu Babu Tolety and Tim A. Majchrzak and quot. 2014, [cit. 2018-02-23]. Dostupné na: https://goo.gl/TLPmoL

[14] DUCKETT, Jon, Gilles RUPPERT a Jack MOORE. JavaScript & jQuery: interactive front-end web development. Indianapolis, IN: Wiley, 2014. ISBN 978-1-118-53164-8.

[15] Introducing Hot Reloading [online]. Martín Bigio 2016, [cit. 2018-02-28]. Dostupné na: https://facebook.github.io/react-native/blog/2016/03/24/introducing-hot-reloading.html

[16] iOS Distribution and iOS Market Share [online]. 2018 Apteligent, Inc, [cit. 2018-03-05]. Dostupné na: https://data.apteligent.com/ios/

[17] Android version market share distribution among smartphone owners as of September 2017 [online]. Statista 2018, [cit. 2018-03-05]. Dostupné na: https://www.statista.com/statistics/271774/share-of-android-platforms-on-mobile-devices-with-android-os/

[18] From Photon to the future: A not-so-brief history of Windows Phone [online]. Andy Weir 2016, [cit. 2018-03-05]. Dostupné na: https://www.neowin.net/news/from-photon-to-the-future-a-not-so-brief-history-of-windows-phone

[19] Comparing five popular frameworks for mobile development in 2017 [online]. Vladimir Kuznetsov 2017, [cit. 2018-03-06]. Dostupné na: https://www.graph.uk/blog/mobile-development-frameworks-in-2016

[20] WARGO, John M. PhoneGap essentials: building cross-platform mobile apps. Upper Saddle River, NJ: Addison-Wesley, c2012. ISBN 978-0-321-81429-6.

[21] A brief history of React Native [online]. Robert Sekulić 2016, [cit. 2018-03-6]. Dostupné na: https://medium.com/react-native-development/a-brief-history-of-react-native-aae11f4ca39

[22] NOVICK, Vladimir. React Native–building mobile apps with Javascript : build real-world iOS and Android native apps with JavaScript. Birmingham, UK: Packt Publishing, 2017. Print.

[23] Introduction to Custom Renderers [online]. David Britch 2016, [cit. 2018-03-06]. Dostupné na: https://docs.microsoft.com/cs-cz/xamarin/xamarin-forms/app-fundamentals/custom-renderer/introduction

[24] Xamarin [online]. Wikipedia: the free encyclopedia [online]. San Francisco (CA), [cit. 2018-03-8]. Dostupné na: https://en.wikipedia.org/wiki/Xamarin

[25] SHACKLES, Greg. Mobile development with C#. Sebastopol, CA: O'Reilly, c2012. ISBN 978-1-449-32023-2.

[26] History Of Virtual Reality [online]. Virtual Reality Society 2017, [cit. 2018-03-10]. Dostupné na: https://www.vrs.org.uk/virtual-reality/history.html

[27] Native Modules [online]. Facebook Inc. 2018, [cit. 2018-03-011]. Dostupné na: https://facebook.github.io/react-native/docs/native-modules-ios.html

[28] Introduction [online]. © 2005-2018 Mozilla and individual contributors, [cit. 2018-03-12]. Dostupné na: https://developer.mozilla.org/en-US/docs/Web/JavaScript/Guide/Introduction

[29] Distance on a sphere: The Haversine Formula [online]. GeoNet cotributors 2017, [cit. 2018-03-12]. Dostupné: https://goo.gl/MnXseJ

[30] JavaScript? Babel. [online]. Vojtech Miksu 2016, [cit. 2018-03-14]. Dostupné na: https://www.dzejes.cz/babel.html

[31] Human Interface Guidelines. [online]. Apple 2018, [cit. 2018-03-18]. Dostupné na: https://developer.apple.com/ios/human-interface-guidelines/overview/themes/

[32] Structure and Data Flow [online]. Facebook Inc 2014-2015 , [cit. 2018-03-19]. Dostupné na: https://facebook.github.io/flux/docs/in-depth-overview.html

[33] Field of View [online]. PanoTools.org 2011, [cit. 2018-03-19]. Dostupné na: https://goo.gl/mFfSgQ

[34] Understanding the React Component Lifecycle [online]. A. Sharif 2015, [cit. 2018-03-25]. Dostupné na: http://busypeoples.github.io/post/react-component-lifecycle/

A Instalace PhoneGap

Nástroj PhoneGap můžeme nainstalovat jako desktopovou aplikaci nebo jako CLI aplikaci. Desktopová aplikace nabízí jednoduché grafické rozhraní, ale oproti PhoneGap CLI nemá takové množství funkcí. V pokročilejší fázi vývoje jsou potřebná kompilace projektu pro vybranou platformu.

Pro samotnou instalaci PhoneGap CLI je nutné mít nainstalovány prerekvizity Node.js a git. Jak bylo uvedeno na začátku práce, Node.js rozšíření JavaSciptu pro tvorbu serverových aplikací. Git je distribuovaný systém pro správu verzí zdrojových kódů.

Instalace Android SDK probíhá spolu s instalací Android Studia a pro testovací účely je výhodou přidat i HAXM Intel a Android Virtual Device pro simulování běhu na různých mobilních zařízení. Poté je potřeba nastavit globální proměnnou prostředí *ANDROID_HOME*, která specifikuje cestu k Android SDK.

Instalace prostředí pro vývoj iOS aplikací je podstatně jednodušší, jelikož je zapotřebí pouze nainstalovaný Xcode.

Samotnou instalaci PhoneGap CLI lze spustit pomocí terminálu nebo příkazového řádku přes *npm*. V podstatě se jedná o manažer balíčků pro prostředí Node.js, který má online databázi možných balíčků. Uživatel poté přes klienta může balíčky stahovat.

```
npm install -g phonegap@latest
```

Po stažení a nainstalování se můžeme po správnosti instalace přesvědčit napsáním příkazu *phonegap* do konzole. Měl by nám být zobrazen výpis všech možných příkazů. Níže je uveden přehled těch nejzákladnějších:

```
phonegap create <path>
```

Tímto příkazem lze vytvořit prázdný projekt PhoneGap.

```
phonegap run <platform>
```

Příkaz pro sestavení a spuštění na dané platformě. Lze přidat parametr, jestli se má aplikace spustit na reálném zařízení nebo v simulátoru.

```
phonegap platform [command]
```

Příkazem *platform* lze přidávat nebo odebírat platformy, pro jenž má být projekt sestaven. K dispozici jsou mimo Android a iOS i Amazon Fire OS, Blackberry 10, Firefox OS, Ubuntu a Windows případně Windows Phone 8. Pro uvedené operační systémy poté můžeme najít v dokumentaci návod pro širší nastavení.

```
phonegap serve
```

Tímto příkazem spustí uživatel vývojový sever, na kterém běží balíčkovací manažer. Přes tento server poté mobilní aplikace PhoneGap Developer stahuje zdrojové kódy a umožňuje tak real-time ladění.

```
phonegap plugin [command]
```

Příkazem *plugin* lze přidávat, odebírat pluginy v projektu. Dále tímto příkazem může uživatel vypsat všechny využité pluginy.

```
phonegap template [command]
```

Tento příkaz je užitečný, pouze pokud chceme využít šablonu pro vývoj aplikace. Jsou zde k dispozici šablony, které využívají nějaký front-endový framework například React nebo Framework7.

A.1 PhoneGap Developer

Jedná se o mobilní aplikace, díky níž můžeme provádět ladění. Je volně stažitelná na obchodech Google Store, App store a Microsoft Store. Aplikace funguje jako klient, který přes vývojový sever stáhne zdrojový kód a spustí jej na mobilním telefonu. Nevýhoda je, že aplikace nefunguje, pokud se využívají nějaké pluginy pracující s nativní API. Pro takový účel musí být aplikace sestavena a spuštěna v simulátoru nebo na reálném zařízení.

B Instalace React Native

Instalace probíhá přes *npm*, tudíž musí být v systému nainstalován Node.js. Instalace Node.js je popsána v příloze A. React Native používá při vývoji aplikace pouze rozhraní terminálu, které nainstalujeme obdobným příkazem jako PhoneGap CLI.

```
npm install -g react-native-cli
```

Po spuštění příkazu a nainstalování React Native CLI, je nyní možné pomocí příkazu níže založit projekt. O všechno se postará *npm*, které si stáhne poslední verzi React Native a všechny potřebné knihovny. Dále lze přidávat různé parametry měnící například verzi frameworku nebo předem připravenou šablonu projektu.

```
react-native init <name>
```

Také je možné založit projekt pomocí dalšího příkazu, který ke svému spuštění potřebuje nástroj create-react-native-app. Tato distribuce React Native přichází s částečnou podporou již zmiňované sady Expo.

```
create-react-native-app init <name>
```

Příkazy uvedené níže spustí aplikace pro danou platformu. Pokud máme připojené zařízení pro ladění, aplikace se primárně spustí na zařízení, pokud to lze provést. Pokud tato možnost není spuštění probíhá v simulátoru.

```
react-native run-ios
react-native run-android
```

Přidání různých knihoven probíhá taktéž přes *npm*. Ten následujícím příkazem uloží a poté nalinkuje do projektu uvedenou knihovnu.

```
npm install <name> --save
react-native link
```

B.0.1 Nastavení prostředí

Pro sestavení a spuštění aplikace pro Android musel být do složky s vytvořenou aplikace přidán soubor *local.properties*. Tento soubor obsahuje řádek s cestou k Android SDK.

```
sdk.dir = /Users/username/Library/Android/sdk
```

C Příloha DVD-ROM

K diplomové práci je přiložen DVD-ROM obsahující veškeré zdrojové kódy a obrázky aplikací.